# UNIVERSIDADE NOVA DE LISBOA

**Faculdade de Ciências e Tecnologia**

**Departamento de Engenharia Electrotécnica e Computadores**


# Diagnosis of an EPS Module


João Paulo de Sousa Ferreira


Dissertação apresentada na Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa para obtenção do grau de Mestre em Engenharia Electrotécnica e Computadores

**Orientador:** Prof. José Barata

Lisboa

2010

# UNIVERSIDADE NOVA DE LISBOA

**Faculty of Sciences and Technology**

**Department of Electrical and Computer Engineering**


# Diagnosis of an EPS Module


João Paulo de Sousa Ferreira


Dissertation presented in the Faculty of Sciences and Technology of the New University of Lisbon to obtain the Master degree in Electrical and Computer Engineering

**Supervisor:** Prof. José Barata

Lisbon

2010

*To my Father, my Mother, my Sister and to Ana*

# ACKNOWLEDGMENTS

This work marks the end of a journey that started in 2004. Throughout the last six years studying Electrical and Computers Engineering I had the privilege to live and share great moments with people that I will always remember.

A special recognition should be made to Prof. Luis Ribeiro. He was a truly instigator and motivator of this work and all my achievements during this last year I have to thank him. Every moment shared with him has been an honor to me, both professionally and personally. His professionalism, dedication, personality and willing to go further will always be an inspiration for me. Nothing that I would possibly say would be enough to thank him for his friendship and belief in me.

I must also thank to Prof. José Barata for being my supervisor and for the possibility of developing such interesting work as well as for all the opportunities that he had provided me along this year.

I could not fail to mention and thank my lab colleagues Bruno Alves and Pedro Barreira and all my friends who accompanied me in this journey.

A word of gratitude goes to Josélia, the person that had the hard task of teach me English and have always encouraged me to go further. To her family, Pedro and Carlos I am also thankful for their friendship.

Another person that I owe a big recognition is Fabio Santos. Nothing that I can do will ever thank him for is friendship and brotherhood. He is a friend that always gives, expecting nothing in

return. I am grateful for all these years that we were roommates and all the experiences that we shared with each other.

For last I left my family and girlfriend Ana, not because they are less important, but because they are the persons that made me who I am. They not only have provided me with the economical means to complete this journey, but most importantly they have loved me unconditionally and taught me the principles of life that I have today. They are my ultimate example. I must reinforce my gratitude to my father, mother, sister, Ana and my aunts Emilia and Celeste because they are the reason of my success

.

# RESUMO

Esta tese aborda e contextualiza a problemática da realização de diagnóstico de Evolvable Production Systems (EPS). Um sistema EPS é uma entidade complexa e animada, composta por modulos inteligentes que comunicam através de mecanismos biologicamente inspirados, para garantir disponibilidade do sistema e capacidade de reconfiguração.

A actual conjuntura económica juntamente com o aumento da procura de produtos personalizados de alta qualidade e a baixo custo, impuseram uma mudança de políticas de produção nas empresas. Neste sentido os mecanismos de produção têm de ser mais ágeis e flexíveis, de modo a acomodarem os novos paradigmas de produção. Ao invés de serem vendedoras de produtos, as empresas estão cada vez mais a assumir-se como vendedoras de serviços, como forma de explorar novas oportunidades de negócio.

Os novos paradigmas de produção, potenciados pelos avanços das tecnologias de informação (TI), especialmente os standards e tecnologias baseados na Web assim como a progressiva aceitação do conceito de sistemas multiagentes e das tecnologias relacionadas, visam alcançar o desenvolvimento de módulos cujas funções individuais e colectivas se adaptam e evoluem de forma a assegurarem a aptidão e adequação dos sistemas de produção no tratamento de oportunidades de negócio voláteis mas rentáveis. Apesar da riqueza de interacções e do esforço despendido na sua modelação, o potencial de propagação de falhas e de interferência destes ambientes complexos, tem sido ignorado do ponto de vista de diagnóstico.

Com o aumento de componentes autónomos e distribuídos que interagem entre si na execução de projectos, os sistemas de diagnostico actuais tornar-se-ão insuficientes. Apesar de os sistemas dinâmicos actuais serem complexos a até um certo ponto imprevisíveis, a adopção de novas abordagens e tecnologias vem com o custo adicional de um aumento de complexidade.

Enquanto a maioria do esforço de investigação em tais sistemas distribuídos industriais está focado no estudo e estabelecimento de estruturas de controlo, o problema de diagnóstico tem sido relativamente pouco abordado. Há no entanto desafios significativos no diagnóstico de tais sistemas modulares que incluem: a compreensão da propagação de falhas e a garantia de escalabilidade e co-evolução.

O presente trabalho apresenta a implementação de uma arquitectura recente baseada em agentes e orientada às interacções que implementa o conceito EPS e que serve como suporte para a introdução de um novo método de diagnóstico que tem a capacidade de lidar com os desafios dos novos paradigmas de manufactura, e de fornecer uma análise de diagnóstico que explora a dimensão de rede dos sistemas multiagentes.

# ABSTRACT

This thesis addresses and contextualizes the problem of diagnostic of an Evolvable Production System (EPS). An EPS is a complex and lively entity composed of intelligent modules that interact through bio-inspired mechanisms, to ensure high system availability and seamless reconfiguration.

The actual economic situation together with the increasing demand of high quality and low priced customized products imposed a shift in the production policies of enterprises. Shop floors have to become more agile and flexible to accommodate the new production paradigms. Rather than selling products enterprises are establishing a trend of offering services to explore business opportunities.

The new production paradigms, potentiated by the advances in Information Technologies (IT), especially in web related standards and technologies as well as the progressive acceptance of the multi-agent systems (MAS) concept and related technologies, envision collections of modules whose individual and collective function adapts and evolves ensuring the fitness and adequacy of the shop floor in tackling profitable but volatile business opportunities. Despite the richness of the interactions and the effort set in modelling them, their potential to favour fault propagation and interference, in these complex environments, has been ignored from a diagnostic point of view.

With the increase of distributed and autonomous components that interact in the execution of processes current diagnostic approaches will soon be insufficient. While current system dynamics are complex and to a certain extent unpredictable the adoption of the next generation of approaches and technologies comes at the cost of a yet increased complexity.

Whereas most of the research in such distributed industrial systems is focused in the study and establishment of control structures, the problem of diagnosis has been left relatively unattended. There are however significant open challenges in the diagnosis of such modular systems including: understanding fault propagation and ensuring scalability and co-evolution.

This work provides an implementation of a state-of-the-art agent-based interaction-oriented architecture compliant with the EPS paradigm that supports the introduction of a new developed diagnostic algorithm that has the ability to cope with the modern manufacturing paradigm challenges and to provide diagnostic analysis that explores the network dimension of multi-agent systems.

# GLOSSARY OF ABRAVIATIONS

| | |
|---|---|
| EPS | Evolvable Production System |
| EAS | Evolvable Assembly System |
| MAS | Multi-Agent System |
| HMM | Hidden Markov Models |
| FMS | Flexible Manufacturing System |
| AMS | Agile Manufacturing System |
| ICT | Information and Communication technology |
| HMS | Holonic Manufacturing System |
| RMS | Reconfigurable Manufacturing System |
| AI | Artificial Intelligence |
| IT | Information Technologies |
| IFAC | International Federation of Automatic Control |
| FDI | Fault Detection and Isolation |
| EKF | Extended Kalman Filter |
| SDG | Signed Digraph |
| FTA | Fault Tree analysis |
| QPA | Qualitative Process Automation |
| PAH | Probabilistic Abstraction Hierarchy |
| CPM | Class-specific Probabilistic Model |
| QTA | Qualitative Trend analysis |
| PCA | Principal Component Analysis |

| | |
|---|---|
| ANN | Artificial Neural Networks |
| SOA | Service Oriented Architecture |
| DPWS | Devices Profile for Web Services |

# TABLE OF CONTENTS

# LIST OF CHARTS

# LIST OF TABLES

# LIST OF FIGURES

# 1 INTRODUCTION

*This chapter introduces the research problem which based and substantiated the implementation of this thesis. Moreover a brief description and summarization of the organization of the document is presented.*

## 1.1 Research Problem

The recent socio-economic crisis along with the normal unpredictability of the business environment challenges the emergence of new production trends and paradigms to cope with these highly dynamic, unpredictable and demanding economies. Therefore, diagnostic is becoming a crucial pillar in enterprises sustainability.



Through time the market requirements are becoming more demanding. Small and medium size industries tend to focus their target customers and core business as a strategy to deal with market competitiveness. Enterprises have to be increasingly prepared to quickly respond to sporadic market opportunities. Near zero device downtime, production quality, low maintenance costs are now more important than ever, each minute with suspended production may be synonym of thousands of euros of loss.

Environmental awareness is another important precursor of production changes. Industries are known to be one of the majors contributors to the carbon footprint [2], therefore sustainable production is a must, in order to minimize industrial debris and optimize the utilization of the finite natural resources. Moreover, industrial accidents present a major risk

to unique natural habitats and also to society. Even though major accidents are the most noticed, the medium to smaller accidents can represent a loss of millions [1].

Moreover mass customization, where each customer demands a personalized product, is nowadays a reality. This new customer's demand shortened the products life cycles and increased the production requirements.

To address this scenarios research and development of distributed production paradigms have increased through time achieving nowadays a very significant importance. Its importance can be verified through the number of related projects from the European Framework Programmes (FP) and from other similar programs.

From FP4 the project DIAMOND (Distributed Architecture for Monitoring and Diagnosis) [3]. In FP5 DEPAUDE (Dependability for embedded automation systems in dynamic environments with intra-site and inter-site distribution aspects) [4] among others. More recently SODA, a Schneider Electric's effort in this area, using Service Oriented Architecture (SOA) [5], and SOCRADES (Service-oriented cross-layer infrastructure for distributed smart embedded systems) [6] developed under the project SIRENA (Service Infrastructure for Real-time Embedded Networked Devices) [7], EUPASS (Evolvable Ultra-Precision Assembly Systems) [8], developed under the FP6. Disc (Distributed Supervisory Control of Large Plants) [9], IDEAS (Instantly Deployable Evolvable Assembly Systems), and SELFLEARNING (Reliable Self-learning Production Systems based on Context Aware Services). These are just a small sample of the research efforts in this area.



Diagnostic plays a crucial role in these systems, regarding the efficiency of the manufacturing system and the durability of the hardware equipment, which on average represents a huge initial investment. Furthermore the desirability of removing the human being from the manufacturing processes, except for maintenance actions, demands the development of self-diagnosable systems as well as self-maintenance and other self-* characteristics. These characteristics are only possible with distributed architectures.

Distributed architectures are usually complex systems and consequently more demanding to diagnose, consequently classical diagnostic approaches are not usually prepared to cope with these new scenarios. Therefore a new distributed interaction-based diagnostic approach compliant with the complexity of distributed state-of-the-art architectures is presented. As base for a distributed diagnostic system the developed work also presents an agent-based interaction-oriented distributed architecture. Relying on generic device abstracting agents, the architecture is characterized by the high level of granularity, self-* capabilities and emergence of skills. The diagnostic system not only counts on the sensor information, but also uses the richness of interaction of complex systems to reason about the agent state. The diagnostic algorithm has the ability to co-evolve with the remaining system, through learning and adaptation to the operational conditions. Moreover, the idea is to foster the emergence of global diagnostic perspective through local diagnostic, which allow the perception of the fault evolution within the network.



The architecture is Multi-Agent System (MAS) based and uses the platform Java Agent Development Framework (JADE) implemented in the JAVA programming language. The agent technology was selected foremost because MAS architectures are the fundamental engines underlying components autonomy that support the implementation of behaviours, dynamic and open environments. Agents provide a proper way to consider complex system with multiple distinct and independent components. JADE is a framework FIPA compliant and can be distributed across various machines [10, 11]. Moreover the agent technology meets all the performance requirements of state-of-the-art architectures, as well as modern production paradigms.

The diagnostic algorithm is supported through a Hidden Markov Models (HMM) which is a statistical Markov model with unobserved states. The HMM learning capacity along with other fundamental assumptions and capacity of some particular problems resolutions make the HMM a suitable tool to implement a distributed diagnostic system.

## 1.2 Thesis Presentation

This thesis is organized in five main chapters: Introduction, State-of-the-art review, A diagnostic infrastructure for EPS, Implementation and Conclusion and future work.

The Introduction chapter briefly substantiates the research problem and describes the organization of the document.

The second chapter on State-of-the-art contextualizes the developed work within the existing manufacturing paradigms as well as surveys the classical diagnostic methods and discusses its importance in nowadays markets and economic environment. Moreover some supporting theoretical concepts on diagnosis, HMM and the philosophical approach of Evolvable Production (EPS) are presented.



The third chapter "A diagnostic infrastructure for EPS" details the hardware infrastructure, as well as presents the system and diagnosis architecture based on EPS paradigm. This chapter outlines the singularities of this work, and exposes the choices made to accomplish the desired system and architecture.

The fourth chapter presents the implementation that validates the architecture introduced in the third chapter. Each individual participating entity is fully detailed. The control system, communications between agents along with the diagnostic systems… are exhaustively described, as well as the system interface is briefly depicted. The validation process is also explained, through the clarification of the testing scenarios and the presentation of the achieved results.

The fifth and last chapter points out the obtained results and contributions achieved with the completion of this work. Future work and research directions are also proposed.

# 2 STATE-OF-THE-ART REVIEW

*In this chapter a review on the state-of-the-art of manufacturing paradigms, diagnostic in distributed and manufacturing techniques is presented along with an overview on Hidden Markov Models and random graphs analysis.*

## 2.1 Manufacturing Paradigms

The first references to manufacturing flexibility report to Diebol (1952). Diebol recognized flexibility as an essential part for medium and short-run manufacturing of discrete parts [12]. In the past, the manufacturing world was ruled by the economy of scales. Mass production and full utilization of plant resources were the way to maximize profit. This philosophy led to inflexible and not easily reconfigurable shop floors.



Manufacturing industry must accompany society's demands. Therefore manufacturing paradigms are obliged to evolve accordingly. In the 70s during the mass production period workers started to become more aware of their working conditions, and customers more demanding regarding the variety and quality of products. The response for the new demands was the appearance of flexible manufacturing system (FMS) in the early 70s. A FMS relates a "manufacturing system capable of automatically performing a plurality of machining operations on each of a plurality of unfinished parts, and more specifically, to a control apparatus for regulating the operation of such a manufacturing system." [13]. Since 1980 in pursuit of greater

flexibility, elimination of inventory excess, shortened lead-times, and higher level of quality, industrial analysts popularized the term lean manufacturing. In the early 90s a study was carried out to study how US industrial competiveness would or might evolve during the next 15 years. One of the study results was the introduction of the agile manufacturing system (AMS) concept [14].

Similarly in the early 90s other paradigms emerged. Lean manufacturing system is one that meets high throughput or service demands with limited supplies, and with minimal waste. The main repercussion of lean manufacturing is the way that production is organized and managed. It influences the all production process, from the suppliers to the customers. The most important idea behind lean manufacturing is avoiding waste [15].

Agile manufacturing is a step further relating lean manufacturing, since lean manufacturing deals with things that cannot be controlled [16]. Agility influences different areas of manufacturing, from management to shop floor. The agile manufacturing desire the integration of design, engineering, and manufacturing with marketing and sales, which can only be achieved with information and communication technology (ICT) [15]. Agile manufacturing is a response to complexity caused by the constant changing's, whereas lean manufacturing is a response to competitive pressure with limited resources.



In the early 90s the use of the holonic concept in the design of manufacturing system emerged [17]. The Holonic Manufacturing Systems (HMS) [18, 19] is a paradigm that reproduces in manufacturing the concepts developed by Arthur Koestler to living organism and social organization. The holon represents the basic unit of organization in living organisms and social organizations. A holon can represent a physical or logical activity, such as a robot, a machine, an order, a flexible manufacturing system or a human operator. It is possible to divide a holon in a set of holons and each one of that holons in another set, and so on. Holons can be implemented through agents. A holon can simultaneously represent a whole and a part of the whole [20].

In the last decade's globalization has become a reality. The economy is no longer national but global and with it, new market demands have emerged. Society wants high quality products at low prices, highly personalized which implies short life cycles. To respond to this new market demands, enterprises had to start focusing their core business [21]. Though, in order to respond to short-term business opportunities enterprises had to start develop networking, to share knowledge and form partnerships [22]. The concept of reconfigurable manufacturing system (RMS) was introduced [23, 24] to respond to this new market oriented manufacturing environment [25]. A RMS is designed to conciliate both productivity and ability to react to changes. RMS capacity and functionality can change over time while the system reacts to new market circumstances. As described in [23] "A Reconfigurable Manufacturing System (RMS) is designed at the outset for rapid change in structure, as well as in hardware and software components, in order to quickly adjust production capacity and functionality within a part family in response to sudden changes in market or in regulatory requirements."



On a study of Manufacturing 2020 [26], the RMS concept was identified as the number one priority technology for future manufacturing and one of the six key research challenges. A survey was also conducted by experts in 1997 [27], which tried to explain the experiences to date with FMS and identifies their accomplishment and failure [25].

The traditional approaches used to implement manufacturing planning, scheduling and control are mostly centralized, which limits the expandability and reconfiguration capabilities of the manufacturing systems. Hierarchical approaches force the establishment of dependencies, where information is processed sequentially by a centralized software supervisor [28]. Multi-agent systems are prone to failures typical of any distributed system [29]. An agent architecture conceptualize an agent as a independent reactive/proactive entity [30]. Each agent has perception, action and reasoning capabilities. These types of architectures facilitate agent's interactions under environmental constraints, and allow the agents to take advantage of cooperation and social interactions. Moreover one of the current factors fostering the development of MAS architectures is the internet which provides the basis for an open environment where agents interact with each other's to reach their individual or shared goals [30].

Agent technology provides a natural way to implement and design distributed manufacturing systems. Therefore MAS architectures are a appropriated technology to implement modern manufacturing paradigms. CoBASA [15] and ADACOR [31] are two examples of recent manufacturing paradigms implemented through MAS architecture. CoBASA is a multi-agent based reference architecture that support fast adaptation and changes of shop floor control architecture with minimal effort [15]. ADACOR is an agile and adaptive manufacturing control architecture that increases agility and flexibility of enterprises, dealing with the need for the fast reaction to disturbances at the shop floor level [31]. In [32] a complete review in agent-based systems to manufacturing is presented.

### 2.1.1 Evolvable Production Systems

Evolvable production System (EPS) or Evolvable Assembly System (EAS) is a concept very related with RMS [33, 34].



There are two fundamental guiding principles in EAS/EPS [35-37]:

- "Principle 1: the most innovative product design can only be achieved if no assembly process constraints are posed. The ensuing, fully independent, process selection procedure may then result in an optimal assembly methodology."
- "Principle 2: Systems under a dynamic condition need to be evolvable, i.e., they need to have an inherent capability of evolution to address the new or changing set of requirements."

As major differences between EPS and RMS, it is important to stress that EPS focuses on adaptability of components as well as on re-engineering needs of the assembly system. Regarding modularity, EPS present a much higher level of granularity, since the abstraction unit can be the desired one. It also ensures the system robustness when the system is facing disturbances.

EPS is a biologically inspired paradigm, nevertheless other sources of inspiration can be highlighted, once they provide helpful concepts, ideas and theoretical background, such as

artificial intelligence (AI), complexity theory, system theory and cybernetics, artificial life including swarm theory and mobile robots and autonomic computing, to cope with complexity [38].

EPS systems rely on many simple, decoupled, intelligent, proactive and re-configurable modules or entities, which through self-capabilities or physical module addition allow evolution of the manufacturing system. These represent the main requisites for pluggable, evolvable and reconfigurable system [39]. Evolvability implies the capacity of co-evolving in line with the changing requirements and the environment. Higher level functionalities emergence is fostered through coalitions and interactions between entities. When considering a group of entities, different elements combinations can originate a variety of higher level functionalities.

In EPS, plug ability is ensured with none necessity of re-programming [34], whereas self-* capabilities concerning installation, management, healing, learning, diagnostic and other features are used to increase the system autonomy and minimize user interaction. One of the requirements for the modern emerging manufacturing paradigms is also making the manufacturing systems more user-friendly [40].



The main characteristics of an EPS system include: distributed control, modularization, intelligent and open architecture as well as a comprehensive and multi-dimensional methodological support that embrace the reference architecture [35]. Moreover, control solutions for EPS must deal with the following aspects:

- Support for integration of modular components, during stationary state and normal production.
- Product changes.
- Fluctuations in demand.
- Provide support for operative phase.

Despite being a concept that still is in research and development, EPS aspire to be a solution to the new market requirements and already proved to be a step further, through a number of implementations presented in the further chapters.

## 2.2 Diagnosis

### 2.2.1 Historical Introduction

In the 60's there was a progressive evolution at the level of goods and services consumption. Associated with this evolution a new demand, regarding the quality price relation, emerged. The need to reduce manpower due to the high salaries, and the search for less monotone jobs, by the employees, were some of the causes to the expansion of process automation.

With the appearance of microprocessors, process automation started to overlap manpower in the production lines. The increased production capacity and the low maintenance costs, compared with the labour costs, allowed broadly higher profits. The development of sensors and actuators, the understanding of the theoretical fundamentals at the automation level, along with the appearance of new information technologies (IT) had a preponderant role in the development and utilization of the diagnostic methodologies. The computer control has brought enormous advances in terms of complex systems. Low level actions performed by specialized operators were successfully replaced by automation control applications. To supervise the autonomous execution and prevent failures, diagnostic methodologies and fault supervision were introduced. A historical survey on diagnostic methodologies development is a multi-disciplinary research area which has to face some literary dispersion [41].



Since early times that the use of diagnostic systems was considered a necessity. Industrial statistics shows that despite low frequency, major industrial disasters have an enormous social, financial and environmental impact. Nevertheless, the medium and small size accidents that originate small injuries and little material damage, cause losses of billions due to their high frequency occurrence [1]. To face this scenario, the control and management of abnormal events, become a priority.

More rigid environmental policies, economic requirements and modern economic competitiveness, reinforce the need of better and more efficient diagnostic systems [1].

Production downtimes and industrial accidents are seen as a source of pollution and waste of energy. These are determinant elements in the lower longevity of expansive equipment.

There are two main areas, through which the diagnostic systems can be developed, hardware redundancy and analytical redundancy. [42]. Hardware redundancy usually relies on a voting system and multiplication of physical devices, in order to detect the location and origin of the fault. This type of diagnostic is however characterized by the high costs that the extra hardware represents. The analytical redundancy uses in turn, relations between system variables through which the system diagnostic is possible.

The early 70s mark the begging of analytical redundancy-based fault research. Beard (1971) developed an observer-based fault detection scheme at MIT. Jones (1973) continued the development of his work [42]. Their work originated a fault detection filter called Bear-Jones. Relatively at the same time statistical approaches to fault diagnosis were first used by Mehra and Peschon (1971) followed by Willsky and Jones (1974). In the year of 1975 Clark, Fosth and Walton used the Luenberger observer for the first time. In 1980 a residual generation scheme based on consistency checking on the system input and output over a time window was introduced by Mironosky [42]. In the late 80s a group of AI researchers proposed a diagnostic method based on First-Order logic. This method utilizes an inference process to identify possible fault components.



Quantitative fault diagnosis methods appeared during the 80s, such as observer-based approaches, parity relation methods and parameter estimation methods among others. In 1991 a Steering Committee called SAFEPROCESS (Fault Detection, Supervision and Safety for Technical Processes) has been created within International Federation of Automatic Control (IFAC). In 1993 SAFEPROCESS became a Technical Committee within IFAC due to its importance [42].

During the last decade, the research focused on fault diagnosis for nonlinear systems. Techniques such as neural networks, fuzzy logic, neuro-fuzzy systems, and genetic algorithms were successfully applied to diagnostic problems [42].

More than ever, diagnostic and fault detection are a very important research field in engineering and particularly in manufacturing systems, since it has implications in a variety of domains like sustainability, efficiency, maintenance and security. Moreover, in critical systems like nuclear plants, aircrafts, submarines and medical devices, among others, the correct and anticipated fault detection is mandatory, otherwise a fault can result in loss of human beings and significant environmental impact [43].

The inevitable development of new technologies and the emergence of new paradigms, led to the appearance of new diagnostic methods, with the purpose of making the system even more productive, efficient and with less and smaller downtimes.

### 2.2.2 Preliminary Concepts



According to Bocaniala and Palade [42] a "fault is the abstraction of an abnormal event that causes unexpected changes in the system normal operating state". A failure represents a serious breakdown of a component or function that implies a serious deviation in the behaviour of the whole system. Although both terms present similarities, a failure is a much more catastrophic event, whereas a fault may not affect the functioning of the overall system.

Isermann in [44] further classifies faults according to time dependency.

- Abrupt fault (stepwise): The fault is felt immediately, and brings the system very close to the limit of acceptable behaviour.
- Incipient fault (drift-like): Its effects increase through time, in the beginning it is almost unnoticeable.
- Intermittent fault: Its effects are noticed in discontinuous periods of time.

A monitoring system used to detect and diagnose fault location along with its relevance to the system is called a fault diagnostic system. Moreover, a fault-tolerant control system is a system which presents operational capacity after the occurrence and recovery of fault events. One

of the important characteristic of this type of systems is automatic reconfiguration when malfunctions are detected and isolated.

A diagnostic system operation can be decomposed into three main tasks, the fault detection which indicates the occurrence or not of a fault, fault isolation, to determine the fault location and fault identification to estimate the fault nature and significance [42].

State-of-the-art methodologies have to face new levels of complexity presented by the distributed characteristic of modern architectures and paradigms [45]. Despite the new challenges faced by modern diagnostic methodologies it is consensual that all diagnostic system should present a number of important requirements [1, 46].

- Quick detection and diagnosis: Concerns the commitment between the quickly faults detection and the system tolerance to noise during normal operation. These two desired characteristics have two conflicting goals.
- Fault isolation: Defines the capability of the system to distinguish between different diagnose failures. This characteristic also presents a commitment that has to be made regarding the isolability and the rejection of modelling uncertainties.
- Robustness: Represents the system capacity to cope with noise and uncertainties, maintaining a reasonable acceptable performance.
- Novelty identifiability: Is the system ability to identify and distinguish between known and unknown failures that might occur. Although the detection of a faulty state is a basic requirement, the identification of unknown events is much more complex once the diagnostic system was not designed to recognize such fault. Is also important to avoid the classification of unknown failures as a known one. Despite the difficulties novelty identification is a desirable characteristic.
- Classification error estimate: In order to ensure user confidence the diagnostic system may provide a priori estimation of the classification of the error that can occur.

- Adaptability: Operation conditions can change not only due to disturbance but also because of environmental conditions, natural detritions and manufacturing requirements. Therefore the diagnostic systems must be adaptable in order to evolve with the changes.
- Explanation facility: Represents the ability to explain the fault origin and path that led the system to the current faulty state. This is an important requirement to keep in mind, once it is through this characteristic that the systems can provide an explanation and assistant to the maintenance operator.
- Modelling requirements: The system must be modelled in order to maximize the diagnostic efficiency, by decreasing the model complexity.
- Computational requirements: Computationally heavy models may affect the system performance. Therefore the diagnostic model should be correctly balanced between complexity and computational power, with the aim of be efficiently and correctly performed.
- Multiple fault identifiability: represents the system capacity of relate and manage different and multiple faults scenarios.



### 2.2.3 Diagnostic Systems Classification

The scientific community, mainly of fault detection and isolation community (FDI) and diagnostic community (artificial intelligent and computer science background), are responsible for the research and development of diagnostic methodologies, is not very clear regarding the classification of diagnostic methods.

According with [47] the diagnostic systems may be categorized according two approaches, regarding the initial knowledge of the system.

- First principles approach: The known information respects the system description, along with the observation of the system behaviour.
- Experiential approach: Methodologies in which the heuristic knowledge plays a key role.

Milne in [48] presents a number of strategies to build diagnostic systems based on knowledge such as structural, behavioural, functional and pattern matching. Knowledge can equally be based on past experiences and acquired information during the process utilization. In this case the classification is shallow, compiled, evidential and process history-based. More information about classification of a priori knowledge can be found in [48, 49].

In [50] diagnostic methods are divided in model-based fault detection methods and fault-diagnosis methods. The model-based fault detections methods are:

- State output observers,
- Parity equations.
- Identification and parameter estimation.
- Bandpass filters.
- Spectral analysis.
- Mean and variance estimation.
- Likelihood-ratio-test, bayes decision.
- Run-sum test, two-probe t-test.



Whereas the fault-diagnosis methods are composed by:

- Geometrical distance and probabilistic methods.
- Artificial neural networks.
- Fuzzy clustering.
- Probabilistic reasoning.
- Possibilistic reasoning with fuzzy logic.
- Reasoning with artificial neural networks.

More recently Venkatasubramanian has proposed a new classification, Figure 1, which can be found in [1, 51, 52].

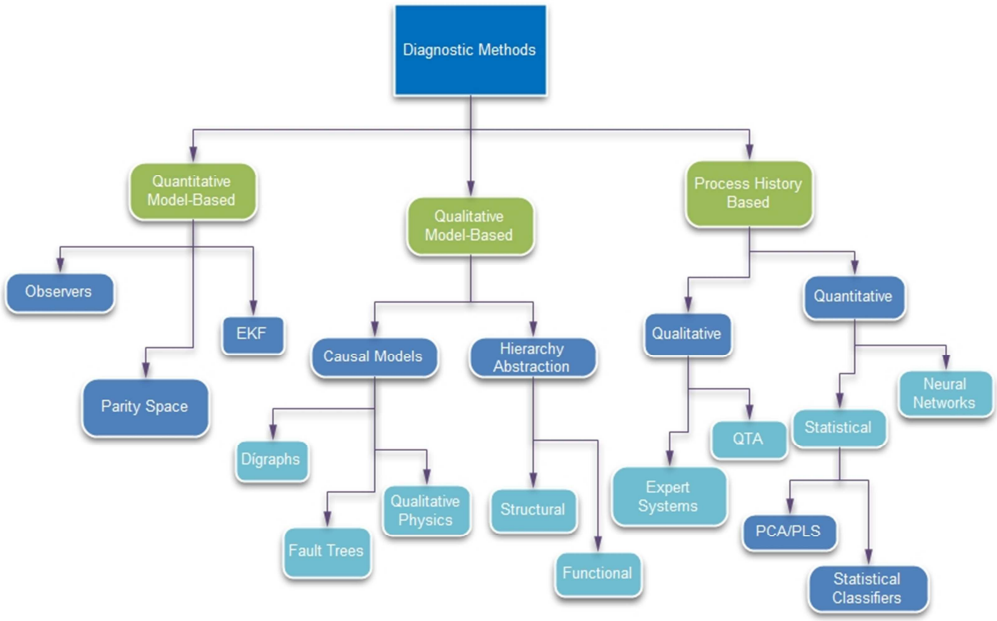


**Figure 1** Diagnostic classification scheme [1].



According to Venkatasubramanian the diagnostic methods can be divided in quantitative methods, qualitative methods and history-based methods. This classification was designed according with two of the main characteristics of the diagnostic systems.

- Type of Knowledge.
- Type of diagnostic search strategy.

The strategy of the diagnostic generation is function of the knowledge representation scheme, which in turn depends on the a priori system information.

Quantitative information model-based relies on information generated from the mathematical relation between the input and the output of the system (residues), Figure 2, which are evaluated by a mathematical function that describes the system.

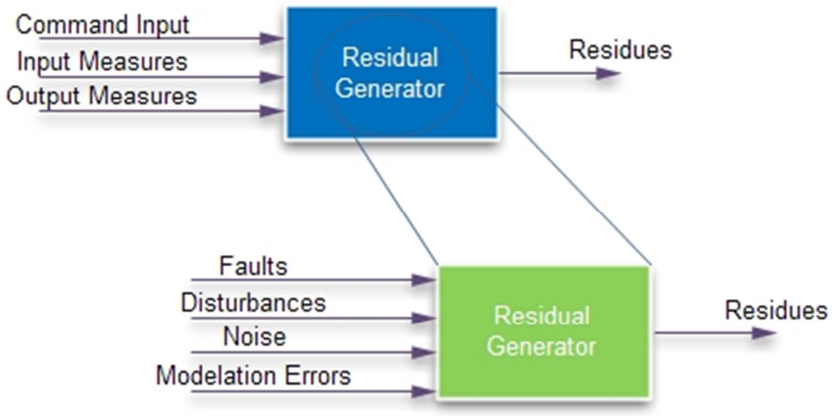


**Figure 2** Residual Generator.



A model-based system is composed by the following main stages [53].

- Residual Generation: computation of input and output of the system to generate residual signal. It uses a system model relating the variables, which is use to check inconsistencies in order to detect faults.
- Decision Making: consist on examine the residuals for fault likelihood and decide if any fault has occurred. Moreover, the correct fault detection is then complemented with the fault isolation procedure to locate the fault.

In qualitative reasoning three different types of reasoning can be addressed. Abductive reasoning implies the generation of explanations for the observations. This type of reasoning is characterized by the number of answers, once for one observation a set of answers can be obtained. The way to choose between answers is by selecting the most probable one. Inductive reasoning is characterized by the establishment of rules categories or concepts that are inferred from a set of representative data. Finally, default reasoning is based in the manipulation of data. It is ideal to use when data need to be updated and override.

Contrasting with model-based approaches, where a prior knowledge of the systems is necessary, in history-based models the requested information is only historical known system information. There are different ways to draw knowledge and build the model through historical information. These methods are used for feature extraction [52]. The development of expert systems based on knowledge, was a first attempt to capturing the knowledge in order to generate conclusions [51].

In this document the author will follow the classification depicted in Figure 1 and [1, 51, 52].

### 2.2.4 Diagnostic Methods Overview

As it was stressed before, diagnostic classification can assume a variety of forms according to the author background and beliefs.



#### 2.2.4.1 Quantitative Model-Based Methods

In the automated control area, fault detection problems are solved through FDI models. All FDI model are two step models, residual generation and decision making, as it was pointed out in a previous section.

Analytical redundancy methods rely on mathematical methods which represent the physical system behaviour. The idea is to compare the behaviour of the real system against the previously created model, and check for inconsistencies [54].

Parity space method is one of the analytical redundancy methods. Historically the parity space technique is related with two distinct sources. The first one is reconciliation and gross error-detection methodology of chemical engineering, while the other source is related with aerospace, through the use of static and after dynamic models, in both space and transfer function [55]. Applications of this method can be found in [56, 57]. For the first example, a case where the systems matrices vary with time, due to changes in the operation point or occasional parameters variations is presented. On the other hand the second example refers to recursive

method for identify parity relation, which are used to build a residue generator based on the parity space. An adaptive monitor method is also presented.

The key in fault detection and isolation through observers is the generation of a set of residues that ensure the correct detection and fault recognition, even in the presence of unknown scenarios. In normal operation mode, all the observers must produce small residues, whereas in the case of a fault occurrence the observer sensitive to the fault should present a much higher residue value. Moreover, each single fault should present a unique pattern in order to allow fault recognition.

A Kalman filter consists of a set of mathematical equations which provide an efficient computational mean to estimate a linear process state, in order to minimize the mean square error. This type of filter enables the estimation of past, actual and future states, even not knowing the precise nature of the modulated system [58]. In [59] a Kalman filter based fault diagnostic system on a networked control system is presented. Kalman filter theory is used to compute filter parameters. Through the obtained values a filter is built and its outputs are used to generate the residues to diagnose the sensor and actuator faults.



If the process or any process related measure is not linear, then an Extended Kalman Filter (EKF) should be used [58]. A use case of an EKF used to detect faults is presented in [60]. The filter is used to generate residues through the comparison between state estimation and real measures that are analysed to detect the occurrence of faults.

### 2.2.4.2 Qualitative Model-Based Methods

Diagnosis concerns the deduction of the structure from the behaviour and for this to be possible, interpretation and reasoning over the recovered data is necessary. This process enables the construction of relations between causes and effects. Those relations can be represented through a Signed Digraph (SDG).

Signed Digraphs are built through nodes and arcs. Each arc is associated with a signal, positive or negative, which represents the effect that a node has on the other. An arc represents a

link between two nodes, which implies the relation between the cause and effect. Each node in the SDG corresponds to the deviation from the steady state of a variable [51].

The diagnostic of multiple failures is a complex problem, once the number of combination might exponentially grow with the number of failures. In [61] SDG based algorithm for multiple fault diagnosis is presented. Knowledge based consisting of knowledge about the process constraints, maintenance schedules among other types of knowledge is used to overcome the low resolution of the SDG. The computational complexity is controlled assuming that the occurrence probability of multiple failure scenarios decreases with the number of failure involved [61]. In [62] a method using SDG to calculate the optimal location of the sensors in order to improve the diagnostic result is introduced.

Fault trees are methods that are usually used to perform analysis of security and confidence in the system. Fault trees are logic trees that propagate primary events or faults up to the higher levels [51]. They are usually built by node layers. Each node is associated with different propagation logic. It is also a good method to prevent or identify failures before they happen, despite the very time consuming and expensive processing in the case of complex trees. Approximations can be used to contour the problem, but they will have repercussions in the precision. Since they use combinatorial logic, fault trees analysis (FTA) is a method that allows fault analysis. Using this methodology, different logic nodes can be applied (OR, AND, XOR), instead of the traditional OR used on Digraphs [63].



Qualitative physics is related with the representation of the physical world, and aims to capture common sense and tacit knowledge used by engineers and scientists. Qualitative representation regards the provided descriptions by the operator or designers of the system. It is important to stress that qualitative knowledge can be derived, even without the possibility of development of a precise mathematical model. These types of models, do not need detailed information which together with the use of symbols facilitates the interaction with the user/expert [64]. Qualitative Process Automation (QPA) is a control method that advocates on-line "control-cycle" generation. This type of representation allows the system to interpret and reason on

heterogeneous data, in order to generate a processing plan in real time [65]. QPA is a methodology based in the cooperation between qualitative physics and expert systems.

Another form of model knowledge is through the development of abstraction hierarchies based on decomposition. This type of decomposition aims the representation of the all system behaviour, through the laws that rule the subsystems. There are some important principals regarding the system decomposition. The non-function-in-structure principle implies that the laws of the subsystems may not depict the whole system. Other important principle is the locality principle that respects the incapacity of a specific part of the system laws to represent other parts of the system.

The system decomposition can be divided in two types:

- Structural: "Specifies the connectivity system information and its subsystems."
- Functional: "Specifies the output of a unit as a function of its inputs."



Diagnostic can be regarded as a top-down search, from the highest level, where the functional systems are considered, to the lower and individual units level, where individual unitary functions are analysed [51].

In [66] a Probabilistic Abstraction Hierarchy (PAH) is introduced. Bayesian networks are used to represent different abstract hierarchies. A PAH model is represented by a tree where each node is associated with a class-specific probabilistic model (CPM). Data is only generated at the leaves of the tree, and a model basically defines a mixture distribution whose components are the CPMs at the leaves of the tree.

### 2.2.4.3 Process History Based Methods

On a daily basis we face situations that in most cases are managed through rules, otherwise the process would be chaotic. A good example is the traffic. Expert systems based on rules are highly specialized and efficient dealing with problems within a restrict domain. An expert system is a computerized system which simulates human expert knowledge in a specific area [67]. These systems are composed by if-then-else rule-base knowledge, which when

necessary are interpreted by the inference engine [52], returning manageable conclusions. The strength of these systems is the knowledge, which at the same time can represent a weakness point whenever unknown conditions are verified. These situations are reported has faults. It is also important to stress that the rule-based knowledge does not represent the physical behaviour of the system.

The main advantages of its use are: an easy development, transparent reasoning, the ability to reason over a some uncertainty and the capacity to provide explanations for the provided solutions [67].

An expert system based on a set covering model is presented in [68]. This type of model is presented as a solution to the difficult problem of multiple and simultaneously faults. In a set covering model the subjacent knowledge to a diagnostic problem is organized according to the representation of disorders and manifestations, where the disorders are symptoms of the manifestations. In [69] a expert system that attempts to address some of the issues involved in bridging the gap between human and computer expertise. The system uses two types of knowledge one based on experience and other based on how the device to be diagnosed works.



Qualitative Trend Analysis (QTA) is a method that can be used to, diagnose and explain the relevant events that occur in the process and to predict future events and states [52]. Qualitative abstraction allows the compact tendency representation through major events. QTA is a method that is performed in two phases, interval-halving trend extraction and semi-quantitative fuzzy trend matching. In the trend extraction phase the signals are extracted as sequences of basic forms, then tendencies extraction is performed. In the trend matching phase, the similarity between two signals is calculated through the evaluation of the likeness of the corresponding tendencies. In [70] a combined method with SDG and QTA for detection of incipient failures is presented. The SDG presents a set of candidate faults based on the incipient response of the process, while the search for the actual fault is then done through the QTA algorithm which uses the temporal evolution of the sensor for further resolution.

Quantitative feature extraction methods-based can be divided in two major methods, statistical feature extraction from process data and neural networks. Within Statistical feature extraction from process data, principal component analysis (PCA) and statistical classifiers will also be referred.

The quantitative approaches essentially formulate the diagnostic problem-solving as a pattern recognition problem. Statistical methods use knowledge from a priori class distributions to perform classification.

Regarding Statistical feature extraction from process data, the future states of stochastic systems cannot be completely determined through passed states, present or future control actions. This limitation is a problem when dealing with systems with random disturbances. The system configuration in a probabilistic manner is a way of dealing with this difficulty. When the system operates in a normal state, the observation presents a distributed probability, otherwise with an abnormal event the system will present other values.



PCA is used to define from an orthogonal partition of the measurement space, two orthogonal sub-spaces: a principal component sub-space and a residual subspace [71]. This method is frequently used when huge and multivariable data sets need to be reduced into a simpler form. In [72] a improvement in a PCA model is presented, through the selection of the personal computer that present the best variable reconstruction. One of the benefits is the à priori fault reduction, caused by non-correlated sensors in the module.

Following a classification approach a diagnostic system can be faced as statistical pattern classifier. The Bayes classifier is an optimal classifier when classes are Gaussian distributions and the distribution information is available. The distance based classifiers use distance metrics to calculate the distance of a given pattern from the means of various classes and classify the pattern to the class from which it is closest. The more basic distance base classifier is the Euclidean classifier. There are also the piecewise and quadratic classifiers [52].

Artificial Neural Networks (ANN) are a computational resource very used in patter recognition. A ANN is a biologically inspired methodology, based on the brain neurons and its

behaviour [73]. An ANN is composed by a group of artificial neurons distributed by layers. The input and output layers represent the system input and output respectively. Moreover the dynamic of the modulated system is represented in the hidden layer. In order to perform the desired task, the ANN is trained with system dynamics representative data. Each ANN can be composed by an infinite number o neurons, however there has to be a compromise between the generalization capacity and the specificity of the network. In [74] a multi-layered feed forward neural-network approach is used to detect and diagnose faults in industrial processes. In this paper particularly it requires a system that is capable of observing multiple data simultaneously. The diagnostic system is basically composed by two-stage neural network. The first detects the dynamic trend of each measurement and the second one detects and diagnoses the faults.

#### 2.2.4.4 Hybrid Methods



A single diagnostic system cannot possess all the desired characteristics. Therefore hybrid systems were developed to integrate more than one diagnostic methodology, which can be quantitative, qualitative or history-based, as a way to suppress the individual diagnostic methods limitations. Some work on hybrid architectures has been developed through the last years. The two-tier approach by Venkatasubramanian and Rich [75] using compiled and model-based knowledge is one of the earliest examples of a hybrid approach [52]. Another hybrid diagnostic system, that relies on Hidden Markov Models (HMM) and dynamic systems models of discrete and continuous time, able to deal with unknown behaviours is introduced in [76]. This capacity allows the diagnostic execution in systems where none assumption is made about the behaviour of one or more components of the system.

### 2.2.5 Diagnosis in Manufacturing and Distributed Systems

Manufacturing systems are usually designed according to the manufacture requirements and necessities, therefore the variety of systems is enormous. For that reasons each system presents a different set of problems to be faced. Since there is not an ideal diagnostic method the same has to be chosen according to the system challenges.

As it was pointed out in a previews section, new manufacturing paradigms are increasingly relying on MAS architectures, due to its distributed characteristics. In this context, a multi-agent based monitoring and diagnostic architecture for industrial components is presented in [77]. In this architecture each component has its own monitoring agent and a semantically distributed group of agents. Moreover a facilitator agent is responsible for the communication between the monitoring agent and the group of diagnostic agents. After processing the data the diagnostic is written in a blackboard. A facilitator agent is informed about the action and checks the blackboard for semantically or spatially conflicting diagnosis. The diagnostic approach uses a model-based combination of semantically and spatially distributed diagnosis and the use of KQML-CORBA (Knowledge Query and Manipulating Language) for implementing the architecture. Similarly a multi-agent diagnostic system where the knowledge is distributed over multiple agents, and each agent possess a model of a subsystem, as a solution to establish global diagnosis of large distributed systems is introduced in [78]. It also relies in a model-based diagnostic where the model knowledge is spatially and semantically distributed over the agents.



A multi-agent distributed intellectualized fault diagnosis system for artillery command system is presented in [79]. Fault diagnosis system based on agent is constructed to realized fault diagnosis on complex system by make use of agent's basic character and multi-agent system's integrated ability. In [80] the handling of sensing failures and work on distributed sensing recovery is presented. Sensor recovery strategies along with considerations about the problems raised by its application in multi-agent platform are discussed. The diagnostic is achieved through a primary data structure EHKS (Exception Handling Knowledge Structure) that is used by the error handler. As a fault is detected a runtime exception is generated and an EHKS is filled with information which will be used for classification and recovery by the error handler.

A diagnostic system with flexibility regarding the information used and diagnoses generated, without sacrifice subject scope or dependency of domain is depicted in [81]. The diagnostic process is organized in a causal model. Each agent contributes with a particular diagnose, with varying levels of precision and complexity. With the production of a diagnostic, the causal model can be used to determine what other diagnoses can enrich the categorization of

the problem. The causal model acts as a sort of map, which allows the diagnostic to progress from easily detectable symptoms to more precise diagnostic hypotheses.

A decoupled and distributed SOA architecture that relies on DPWS technology to abstract the system entities is introduced in [82, 83]. The architecture implements a diagnostic system targeting low cost devices with low processing power and memory, performed through fault related structure knowledge of the domain. Diagnosis at the device level explores ACORDA, a recent prospective logic engine. The ACORDA engine assesses future states and support postponing abductive decisions until the arrival of new data from the realization of experiments.

A web-based multilayer distributed fault diagnosis system for remote diagnosis is discussed in [84]. The described architecture is a data-driven system. Therefore the knowledge learning process extracts expert knowledge from confirmed faults, using various intelligent data mining technologies such as rough set, artificial neural networks and decision trees. The case-based reasoning diagnostic mechanism consist on three main phases: choosing diagnosis parameters, performing diagnosis and saving diagnosis results.



A diagnostic system for heterogeneous manufacturing environments is presented in [85]. The diagnostic knowledge is modelled in terms of causal relation between diagnostic attributes associated with manufacturing processes. The binary causal relations are meshed via their common attributes into a network. An attribute corresponding to the effect in one causal relation can equally correspond to the cause in another causal relation in a uniform manner. A cause-and-effect diagram only depicts the causes of each individual symptom, although the information is complemented through an influence graph, which concerns the relations between symptoms as well as with the relations between causes and other symptoms caused by those same causes. The diagnostic mechanism applies a probabilistic inference method to the influence graph.

A two-levels modelling and control framework for real-time manufacturing processes based on petri nets is introduced in [86]. The system is capable of integrating monitoring and fault detection techniques along with performance optimization procedures. Fault detection is

on-line and is achieved by comparing the actual system response with the optimal control sequences provided by the first order petri net. In the occurrence of a faulty event a diagnostic procedure is triggered and a recovery action is further imposed.

In [87] a combination of three diagnostic methods in order to diagnose completely the operational faults of a manufacturing system is introduced. The diagnostic systems are based on fault tree analysis as well as in logic and sequential control of manufacturing systems by programmable logical controller.

An implementation of an integrated diagnosis system in a manufacturing systems is particularly presented in [88]. The diagnostic is implemented through an expert system. Like in other expert systems, the knowledge can be physical or experimental. In [89] a benchmark manufacturing cell and some experiences of designing supervisors and diagnosis using the Ramage and Wonham framework are discussed. A fault-identification method that is based on nonlinear dynamic model of a robot manipulator is introduced in [90]. A nonlinear observer is used to identify a class of actuators faults after the detection of the fault by other method.



Consensus in a network is a recent approach and can represent an important parameter to account in network distributed diagnosis models. A fault consensus in a network of unmanned vehicles is presented [91]. Each vehicle is equipped with an interactive multiple model algorithm based FDI, as well as a consensus filter. An interactive multiple model based FDI algorithm provides a quantitative measure on the probability of a fault in the network that can be used for fault consensus. A consensus filter in each vehicle generates probability of a current mode of operation using the local probabilities as well as probabilities generated by neighbour's vehicles. In this proposed algorithm, vehicles can detect faults that are not observable through their local measurements.

All the presented methods lead us to conclude that although huge efforts are being taken in the development of new models and architectures, the same are still presenting some of the problems also faced in the more classical approaches. Common to almost all multi-agent approaches is a communication overhead. The diagnostic methodologies are still very system

dependent, which prevents the utilization of the same diagnostic model in other systems. Moreover, the addition of new components, without reprogramming the system is also a feature rarely taken into account along with the fault propagation effects and real time network diagnostic adaptation.

The next section will present requirements and characteristics that must be accomplished when diagnose EPS systems.

### 2.2.6 Diagnosis on EPS

As it was already detailed, future manufacturing paradigms rely in small intelligent devices. The distributed nature of those systems requires an extra attention regarding diagnosis. Despite the distributed and decoupled nature of EPS along with its reliability on self*-capabilities, a diagnostic system for EPS follows the accepted requirements for any similar system. In [46] some requirements are presented such as early fault detection and diagnosis, proper fault discrimination, robustness, identification of multiple faults, fault explanation, adaptability among others.



Each individual intelligent component participating in an EPS architecture is expected to detect and react in real time to faulty scenarios and environmental changes. Self-* capabilities related with diagnostic should be used to perform short, medium and long term analysis. Short terms analysis grants reaction and capture of catastrophic faults, and unexpected events. The purpose of medium and long term analysis is the prediction of future events [39]. Due to the decoupled and distributed nature fault propagation and multiple fault identification, may become a hard problem to tackle [40]. Nevertheless, the communications and interaction capability of the individuals plays a very important role in the resolution of these challenges.

From the architectural and diagnostic point of view, it is possible to foresee two different major challenges that need to be addressed in EPS systems. The diagnostic performed at a low-level, which relies on sensorial information to interpret the device condition, and at a higher level, the capacity to diagnose the system from a network perspective through local information. EPS architectures imply that a network of intelligent modules can abstract from an individual machine

to a full manufacturing plant, according to the desired granularity. It is of major interest not only have individual diagnostic information, but a system diagnostic perspective revealing fault propagation through the network, modules dependencies along with multiple fault origin detection.

A diagnostic system compliant with EPS should also present evolvable capabilities in order to adapt its behaviour according to changes, and preserve functionalities even in scenarios where some part of the diagnostic infrastructure is not working.

## 2.3 Hidden Markov Models

Hidden Markov Models (HMM) were introduced in late 60s and early 70s and were firstly described in a series of statistical papers by Leonard E. Baum and other authors. HMM is very appropriated methodology to perform pattern recognition, particularly speech recognition [92-94]



An HMM [93, 95] is the following triplet:

$$\lambda = (A, B, \pi)$$

where A is a N×N matrix (N is the number of states in the model) denoting the state transition probabilities:

$$P(q_{t+1} = S_j \mid q_t = S_i) = a_{ij}$$

B is an N×M matrix (M is the number of observation symbols) that encloses the observation probabilities:

$$P(O_t = k \mid q_t = S_i) = b_i(k)$$

(i.e. the probability that the observation k happens given the state Si), π represents the starting state probabilities.

In this context, a HMM can be considered a finite state machine where transitions between states are probabilistically driven (the probability of transition to another state given the current state).

When handling HMMs there are three fundamental assumptions [96]:

- The Markov Assumption - the next state is only dependent on the current state and the observations (valid for first order models as the one considered).
- The Stationary Assumption - the state transition probabilities are independent of the time at which they occur.
- The Output Independence Assumption - the current observation is independent of the previous observations.

Additionally HMM support the resolution of three main problems [93]:



- "Problem 1: Given the observation sequence $O = O_1 O_2 \ldots O_T$, and a model $\lambda = (A, B, \pi)$, how do we efficiently compute $P(O|\lambda)$, the probability of the observation sequence, given the model?"
- "Problem 2: Given the observation sequence $O = O_1 O_2 \ldots O_T$, and the model $\lambda$, how do we chose a corresponding state sequence $Q = q_1 q_2 \ldots q_t$ which is optimal in some meaningful sense (i.e. best "explains" the observations) ?"
- "Problem 3: How do we adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(O|\lambda)$?"

The resolution for problem 2 that aims the discovery of the state sequence most likely to be produced by the given observation sequence, can be solve through the Viterbi algorithm, whereas the Baum-Welch algorithm can be used to solve the third problem which is coincident with the implementation of the learning process. Through the insertion of observation sequences the algorithm adjusts the parameters $\lambda$ in order to best describe the observation sequences.

The HMM is a powerful statistical method for modelling generative sequences which can be characterized by an underlying process generating an observable sequence [97]. A HMM is a

doubly embedded stochastic process with an underlying stochastic process that is not observable. An observation represents a probabilistic function of the state. HMM are adequate to represent problems where the internal state is not known and can only be inferred through observations [98]. Therefore and in order to bridge the HMM with diagnostic applications a fault can be considered a hidden state of the system that can be inferred through the observations. Moreover, HMM have the ability to deal with incomplete information [99].

Although not very commonly applied to diagnostic purposes, in [98] a formalization of the diagnostic problem based on HMM is introduced. The implementation showed high accuracy when compared with an optimal solution based on Bayesian inference theory.

## 2.4 Random Graphs

Random graphs were first introduced in the late 50s by Erdós and Rényi through the Erdós-Rényi network. There are two types of construction of random graphs, with a fixed number of vertices [100]:



- Construction Method 1: "Each two vertices of the network are connected by an edge with probability *p*. Naturally, this edge is absent with probability 1-*p*."
- Construction Method 2: "A given number *L* of edges connects randomly chosen pairs of vertices. One can realize this construction procedure by adding new edges one by one and repeatedly connecting randomly chosen pairs of vertices. In graph theory, this is called a *random graph process*."

The number of vertices along with the network connectivity will impact over the network complexity. Complexity usually increases the difficulty in development, testing and maintenance of the systems [101].The first complexity problems were biologically related, since biologically organisms are extremely complex organisms. The necessity of extract more information about this systems led young mathematical biologists to apply theories to assess the information content in the living matter. Around the 80s complexity theory emerged as a new branch of science, due to the spread of highly complex dynamic systems, non-linear dynamics and emergent events [102].

The above section will introduce some basic graph theoretical notions.

## 2.4.1 Adjacency Matrix

The adjacency matrix represents the connections between vertices. Through adjacency matrix it is possible to extract characteristics such as vertex degree ($a_i$) that represents the number of neighbours of a vertex and total adjacency ($A(G)$) which is the sum of all vertex degrees in a graph.

Undirected graphs are represented through a symmetric adjacency matrix, respecting the main diagonal. On the other hand, in direct graph there is no symmetry, moreover the matrix can be divided through the main diagonal in out-degree and in-degree. The main diagonal represents a cycle within the same agent.



$$Adjacency\ matrix\ A_1 = \underbrace{\begin{matrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{matrix}}_{\text{In-degree}} \Big\} \text{ Out-degree}$$

$$a_i = \sum_{j=1}^{V} a_{ij}\,; \qquad A(G) = \sum_{i=1}^{V}\sum_{j=1}^{V} a_{ij} = \sum_{i=1}^{V} a_i$$

Through the adjacency matrix it is also possible to extract some generalized descriptors with respect to network connectivity: the average vertex degree ($< a_i >$) and connectedness ($Conn$).

$$< a_i > = \sum_{j=1}^{V} a_{ij;} \qquad Conn = \frac{A}{V^2} = \frac{2E}{V^2}$$

## 2.4.2 Graph Distances

The adjacency matrix establishes de adjacency between two vertices. A sequence of adjacent edges between two vertices defines a path. According to the network topology, a path can have both senses (undirected graph), or just one sense in the case of a direct graph. A graph

distance matrix indicates the distance between the vertices. The sum of distance between a vertice and all its neighbours it is the vertex distance $d_i$, while $d_{ij}$ defines the distance between two vertices. The sum of all distance matrix entries is called the graph distance, D.

$$d_i = \sum_{j=1}^{V} d_{ij}; \qquad D(G) = \sum_{i=1}^{V}\sum_{j=1}^{V} d_{ij} = \sum_{i=1}^{V} d_i$$

Through the distance matrix it is also possible to compute the average vertex distance (degree) $< d_i >$, and the average path length $< d >$.

$$< d_i > = \frac{D}{V}; \qquad < d >= \frac{D}{V(V-1)}$$

The denominator assumes the value $V^2$when main diagonal values are included.

# 3 A DIAGNOSTIC INFRASTRUCTURE FOR EPS

*This chapter details the implemented distributed intelligent architecture that supports local diagnostic based on the surroundings perception for Evolvable Production Systems.*

## 3.1 Infrastructure Description

In this section the hardware infrastructure, the NOVAFLEX cell, will be particularly depicted.



### 3.1.1 Hardware Infrastructure

#### 3.1.1.1 NOVAFLEX Assembly Cell

The NOVAFLEX assembly cell is composed by the scara station and by the conveyer system. Figure 3 and Figure 4 schematizes the existing hardware.


**Figure 3** NOVAFLEX cell.

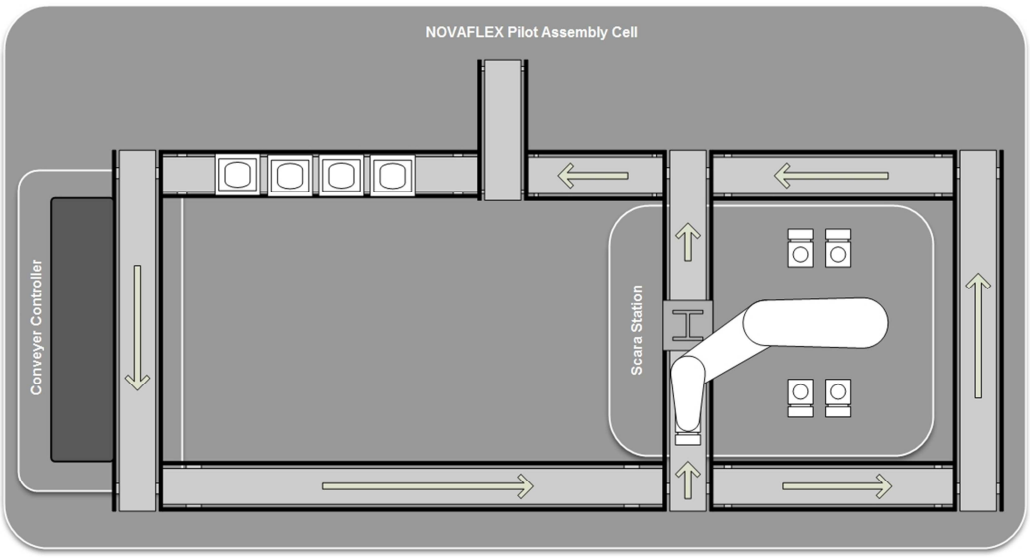


**Figure 4** NOVAFLEX's hardware layout.

### 3.1.1.1.1 Scara Station



The Scara assembly station (Figure 5) is composed by the Scara Robot (Bosch SR800) a gripper warehouse and a fixing equipment.

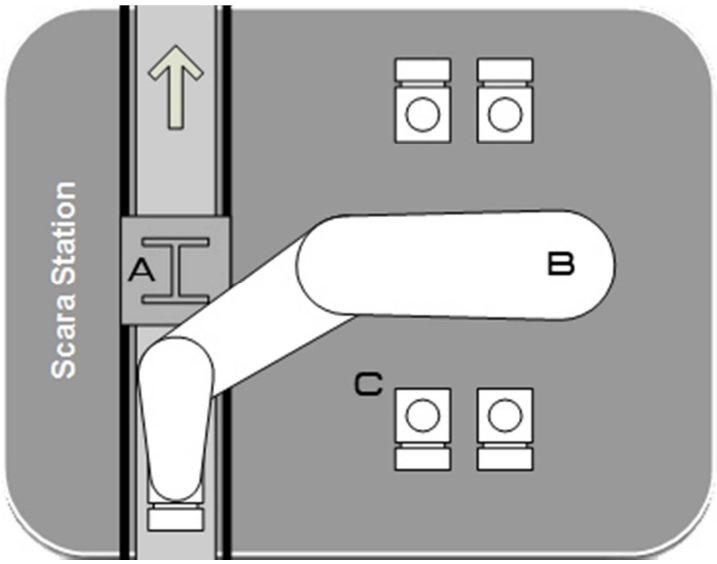


**Figure 5** Scara Station layout.
A – Fixture; B – Scara Robot; C – Tools Warehouse

### 3.1.1.1.2 Conveyer System

The conveyer system is controlled by conveyer controller that is responsible for the control and management of all the conveyers, Figure 6.

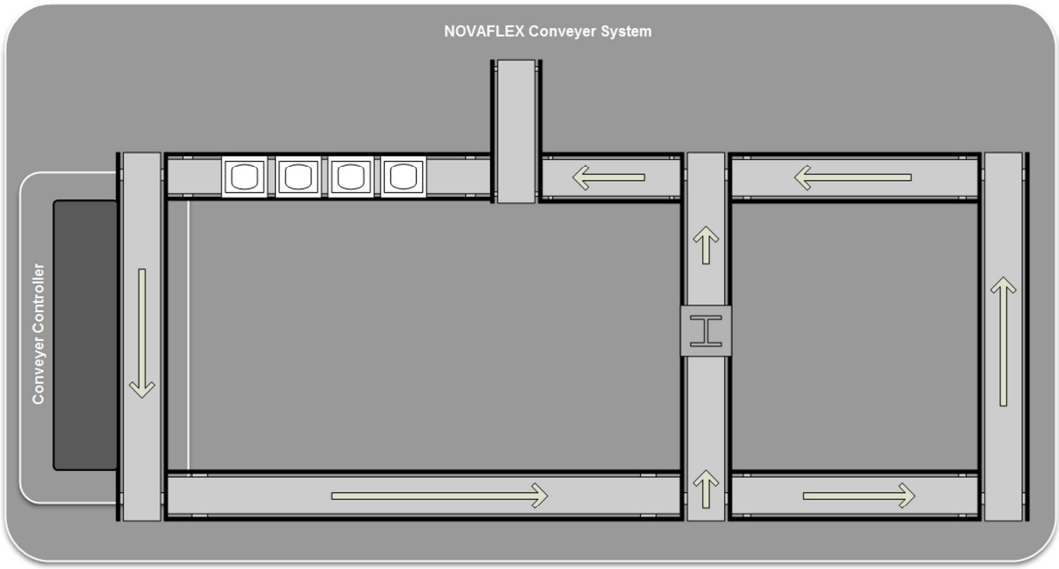


**Figure 6** NOVAFLEX conveyer system layout.

One of the main limitations of the NOVAFLEX conveyer system is the lack of hardware redundancy, making very difficult the resolution of some conveyer faults through the rerouting of the affected pallets.

## 3.2 Hardware Modularity

The crescent ubiquity of computation power along with technological advances is raising new challenges to manufactures.

Some of the actual industrial equipment still presents major constraints at the processing and communicational level, which may restrict the application of innovative control and diagnose schemes. NOVAFLEX legacy base control infrastructure is more aligned with more conservative approaches and does not have the required processing power to support agent base platforms.

Therefore, all the control is pc-based. Each intervenient is abstracted by an agent, notwithstanding the system granularity is domain dependent.

The developed architecture is inspired in some recent works [15] [103] and paradigms in the field of Evolvable Productions Systems [33] [34] [40] which advocates the use of highly granular systems with plug and play characteristics to improve the system scalability and flexibility. Similarly and also under the EPS paradigm MAS and SOA architectures with fine granularity modules are presented in [82, 83, 104, 105]. Another multi-agent based control architecture to support modular reconfigurable production systems is introduced in [103], where the architecture was design in order to allow the addition and removal of new modules in the system, without any system reprogramming.

Classical approaches are by nature more prone to suffer system failures. A failure in the central node usually implies the inoperability of the rest of the system, since the major control and regulatory functions are concentrated in that node. Another drawback is the heavy reprogramming necessity whenever any change need to be applied in the system.



Recent research in this subject tries to avoid these problems through the use of distributed architecture were the intelligence is distributed by each node of the network. Furthermore, new trends impose that rather than reprogrammed the system should be reconfigurable.

Nevertheless the majority of today manufacturing systems are aligned with more conservative approaches, recent works proved that even though legacy systems can successfully adopt recent architectures [15, 83, 106].

In this sense, each intervenient was abstracted by a generic agent that is instantiated according to the component characteristics.

## 3.3 System Architecture

### 3.3.1 Principles and Concepts

Network information technologies evolution has broken many, once insurmountable, barriers. Component intelligence and automation along with environment awareness and adaptation, self-reconfiguration, self-organization, self-learning, self-*, plug ability… will probably be a reality. The research focus of this work was the implementation of an agent-based interaction-oriented platform to support emergent diagnosis [107]. The work was implemented aiming the development of a distributed intelligent infrastructure, as a base for the distributed evolvable self-diagnostic system, capable of support EPS specifications and solve RMS requirements [25], such as modularity, adaptability plug-ability, evolvable capacity among other.

The main challenge of the developed work was the design of a generic infrastructure able to support diagnosis in volatile environments, which at the same time potentiate the system self-* capabilities, evolution, reconfiguration, flexibility and plug ability. The system granularity can be defined in the most convenient way and according to the granularity of the available hardware.



It is important to stress that the system architecture relies and greatly depends on the communication and interactions between agents. Therefore, the social capabilities of each agent foster the emergence of high level skills through the conjugation of multiple local skills as well as potentiate the agent surroundings awareness which will have a major role in the diagnostic mechanism.

Apart the Generic Shop-floor Agent (GSA) there will only be three more agents in the system, the Broker Agent (BA), the Agent Machine Interface (AMI), and the Graph Diagnostic Designer Agent (GDDA). The AMI act has a mediator of interactions between the GSA and the legacy systems. The main purpose of the BA is to be a repository containing all the known skills. The BA keeps track of all existing composed skills relating the conjugation of all possible available resources. Whenever a coalition of agents is formed a set of priority heuristics takes place, in order to decide which agent will be responsible the higher level skill (coalition leader).

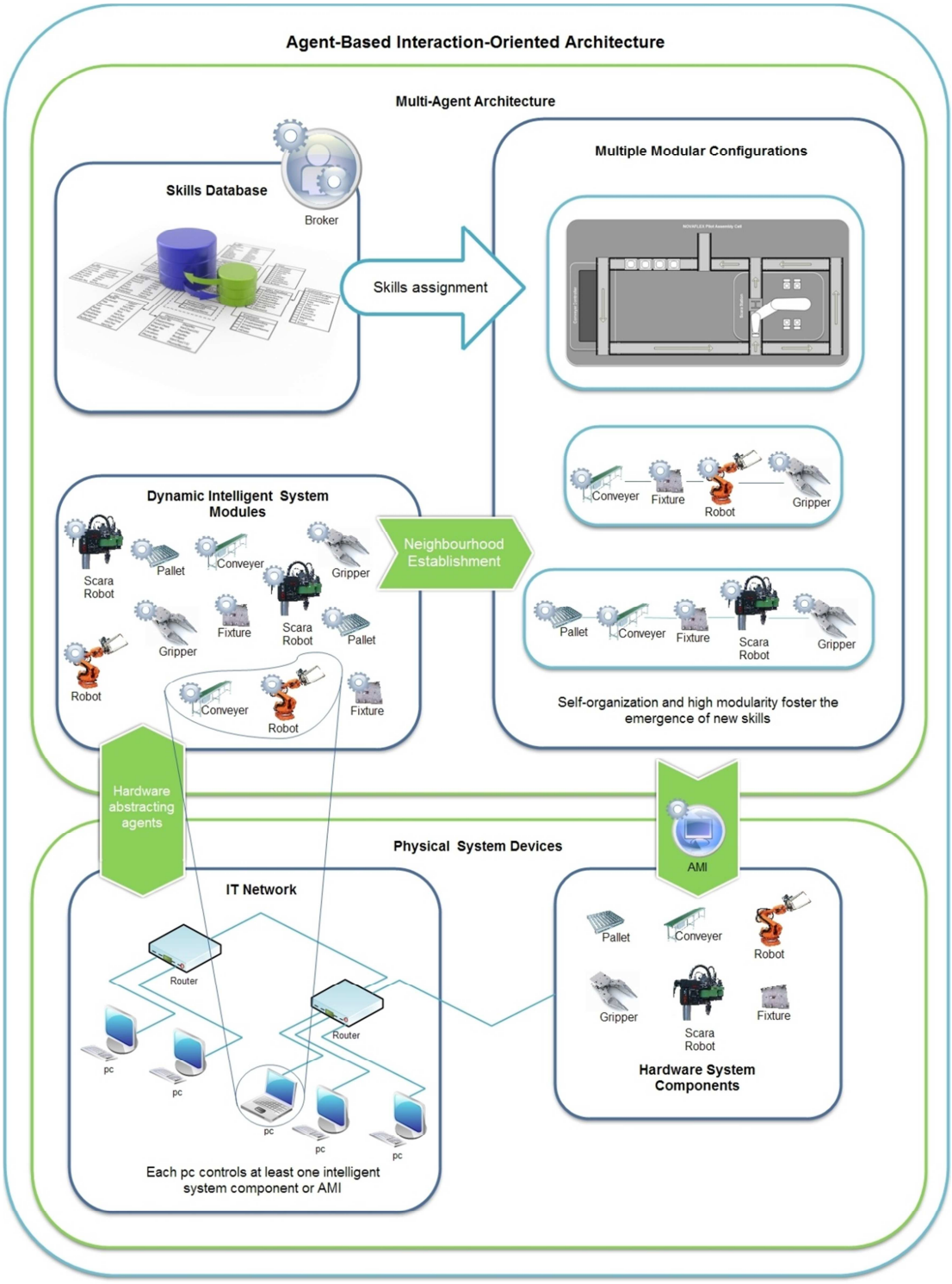


**Figure 7** System Architecture.

The coalition leader is responsible for the execution of the skills involving the coalition agents, as it is responsible for contacting the BA to update the coalition skills. Finally the GDDA is only a graph editor to represent the network layout and the individual diagnostic.

The Figure 7 tries to depict the developed architecture, its modularity and emergent behaviour. The physical system devices represent a possible shop floor configuration, where a network of computers is responsible for the control of the shop floor components. Each computer is at least abstracting an AMI, BA or a GSA. For the first case it is mandatory that the computer has somehow an active connection with the hardware component. Due to the inherent social capacities of the GSA agents, the interactions are established through neighbourhood relations. Whenever an agent establishes a neighbourhood the BA is consulted and if the neighbourhood is able to execute a higher level skill, it will be assigned to the coalition leader. Moreover the execution of any skill from an individual agent or from the coalition leader is sent to the AMI which is responsible to send the order to the hardware component.



Before proceeding with the architecture some concepts need to be introduced.

Self-* system are systems able to manage themselves. Aspects of these systems include properties such as self-awareness, self-organisation, self-configuration, self-management, self-diagnosis, self-correction and self-repair among others. These systems are bio inspired and try to simulate behaviours present in nature [108].

The emergent concept is often used to convey the surprising, inexplicable, or strange nature of a system behaviour. With the increase of complexity it is usual to verify the emergence of unwanted behaviour [109]. In the context of this work, the emergence of skills mean that through the conjugation of resource new capacities, not expected at first sight, appear in the system.

Neighbourhood is the group of agents that are directly linked with an agent. The relation present a generic nature and can be inbound, outbound or both. The neighbourhood concept in this work does not mean physical proximity. Therefore a neighbour can be located in the other side the network. However they have to be linked through an interaction.

Coalition is an important method in a society of agents, to achieve cooperation between agents. Through a coalition an agents can increase its ability to satisfy its own goal, since it has access to resources otherwise inaccessible.

A workflow is a description of a product development project, where the nodes represent the work, and the arrows the flow of control and data between work nodes [110]. In the context of this work a workflow is a logical sequence description of actions, with a defined purpose. Moreover a workflow can present both sequential and parallel actions.

An orchestrator is a module that ensures the execution of specified process plans, distributing actions among distinct agents and reacting according to the feedback provided by different agents on the successful execution of a certain task [111].

In a network, the fault receptivity of each agent varies according to internal and external factors. From natural wear to environmental influences, the causes to malfunction are multiples. Therefore the concept of vulnerability is introduced. A vulnerable agent represents a network node with extremely high fault receptivity [112].



### 3.3.2 Control System

Despite the similarities between the developed architecture and other already referred MAS architectures, with the architecture hereby detailed the authors wanted to take a step further envisioning collections of modules whose individual and collective function adapts and evolves ensuring the fitness and adequacy of the shop floor in tackling profitable but volatile business opportunities. [107]. In order to achieved that goals a new MAS EPS compliant generic control architecture was developed.

The main role is played by the GSA agent, once it represents all the process intervenient entities. The GSA is composed by four major blocks (Figure 8), the agent instantiator, the neighbourhood management, the skills orchestrator and finally the diagnostic system.

The agent instantiator is responsible during the instantiation of the generic agent for the customization process which is defined in an XML description. The neighbourhood management

is the core block supporting the emergent diagnosis. It updates all the agent relations as the agent interacts with other agents (adding, removing, and changing). The skill orchestrator block is responsible for the instantiation, management and execution of skills. These tasks include the validation of preconditions, the dynamic allocation of other agents to fulfil subtasks as well as the

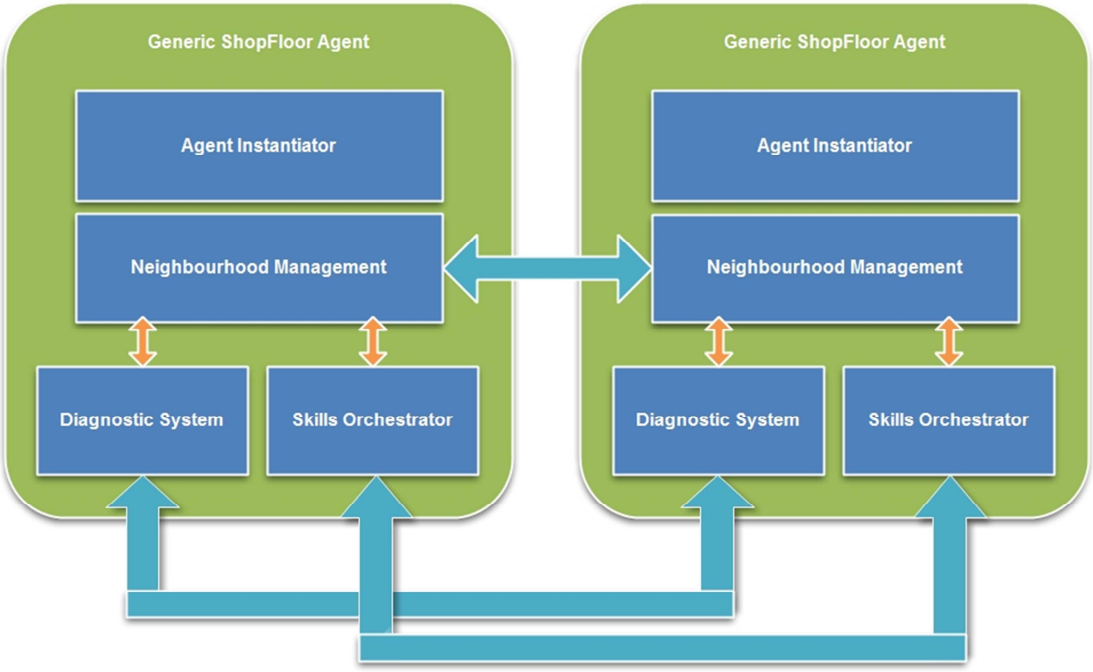


**Figure 8** Generic ShopFloor Agent architecture.



coordination with the neighbourhood management block to ensure a consistent state. Finally, the diagnostic system block is responsible for the diagnostic of the GSA state through sensor and surrounding information.

Although all entities derive from the same structure, some critical parameters are defined in harmony with the abstracted device. This differentiation is performed when an xml user/expert defined file containing the device characteristics is loaded. The load process is not only responsible for the agent characterization but also for the agent skills attribution, according to the functionalities defined in the file. With the purpose of promoting the skills emergence a distinction was established between skills, originating two different types: simple skill and composed skill. A simple skill can be considered the system basic skills, or the skill basic unit that

the agents are able to perform, and therefore offer as a service. A set of simple skills can define a composed skill when correctly aggregated. Moreover a composed skill can be considered as a certain process. After this initial instantiation the GSA has acquired its own basic skills as well as its characteristics. Typically, at the beginning, each independent GSA should only present simple skills which represent the module basic functionalities. There should not be any notion of composed skills or process in the system.

From the architectural point of view, a set of GSAs will only become a system with the establishment of relations between entities. In this sense the system will acquired a network dimension were relations are defined under the neighbourhood concept. This approach maximizes the plug ability and reliability as well as originates failure proof systems due to the high modularity.

The neighbourhood relation has a generic nature. Moreover a neighbour relation is characterized by a type and sense, so in this way any semantic can be associated to it, as long as there is a mean and a purpose for measuring and monitoring it. For example, between a robot and a gripper can be considered that both have a pneumatic relation from the robot to the gripper, since it is the robot that provides air to the gripper. The neighbourhood establishment process is directly related with the skills emergence in the system. Whenever a neighbourhood relation is established a coalition is automatically formed. This behaviour instigates the skills emergence, once the ordered and correct combination of multiple basic skills from different agents can form a process or composed skill. Therefore a hierarchy was defined according to the agent characteristics. Whenever a relation is established the higher ranked agent is responsible for sending a request to the BA in order to get the new coalition skills, and for the management of the emerged skills (Figure 9). Similarly, when a GSA is removed from the network the agent dependent skills will be lost. The more complex is the coalitions, the more complex and diverse can be the composed skills.

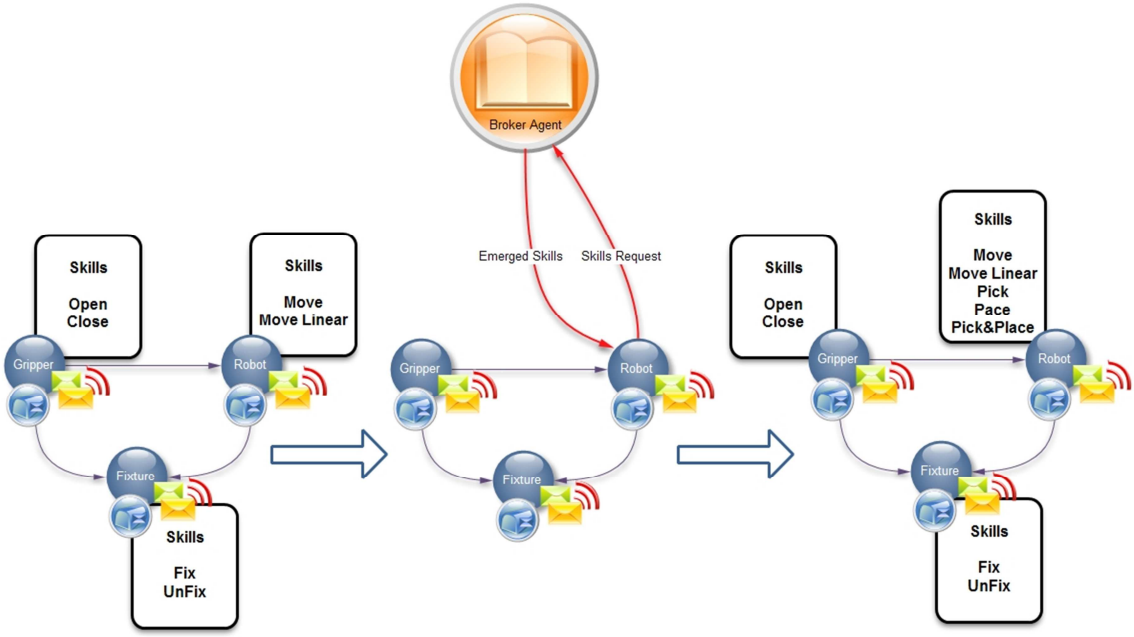


**Figure 9** Broker Agent Request and skill emergence process.

Typically the neighbourhood relations are established by the user/expert during a module addition in the system, nevertheless the workflow manager is also able of autonomously generate interactions with other process modules actors, in order to perform its process plan. Usually the workflow management is played by the unit to be processed, which implies that it has to be aware of its processing plan. The workflow execution starts with the load of a user/expert defined xml file containing all the process steps. Although the workflow management is performed by a single GSA, the workflow execution is distributed over the system according to the workflow specifications. All the GSAs possess a skill orchestrator, responsible for the management of the execution of the attributed skills.



Targeting the recover from faulty scenarios each GSA orchestrator is responsible for finding an alternative neighbour agent to take the place in the execution of the faulty GSA. After trying all the alternative GSAs, an alternative skill can be performed in order to contour the fault and resume the execution. It is important to stress that all the skills can have defined an alternative skill which can be performed, if the execution of the former one fails. The same is valid for the alternative skills. This cycle is performed until there are no more alternative agents or skills. Moreover, if the workflow manager orders another GSA (GSA1 for example) to perform a composed skill, and if in turn the GSA1 orders a GSA2 to execute one of the composing skills, then in the case of a failure, the GSA2 will try to recover from the fault, through the described

method. If the recovery succeeds, the process will be resumed, otherwise the GSA1 will try to recover and so on.

The generic nature of the system implies the use of the same GSA structure independently of the device abstracted. Therefore each GSA abstracting a module has to be aware of which AMI bridges the communication with the corresponding hardware module.

### 3.3.3 Diagnostic System

Usual diagnostic approaches are typically model dependents, or not very complaints with high levels of plug ability, modularity and little or none (re)configuration. Nevertheless EPS follow the consensual [113], and desirable characteristics for industrial applications, expressed in chapter 2.



This work presents a novel approach. Rather than individual diagnostic the goal was the emergence of a global network diagnostic through local information and local intelligence. Each device has to compute over sensorial and surroundings information in order to reason its internal state in a network perspective, which will favor the identification of the fault source as well as its path over the network.

#### 3.3.3.1 Self-Diagnosis

Agents are entities known for their ability to interact and communicate with their peers as a way to cooperatively achieve the expected goals [114]. The developed diagnostic system is based and highly dependent on the neighbour collected information, to correctly reason about its state according to the environment, for that reason a robust and efficient communication system was developed.

At the device level sensors perform a major role in the state evaluation process. Despite their importance, sensors present a variable accuracy rate, depending on its quality as well on external conditions and they are also fallible. In high complex system and high granular systems, an entity is usually in direct contact with many others. Taking advantage of this reality,

surroundings information was included in the diagnostic process was a way to strength the diagnostic confidence level and to account the fault evolution in the network.

The diagnostic mechanism is supported by an HMM model. Moreover the GSA internal state can be represented through five hidden previously defined states, presented in Table 1.

| State | Description |
|---|---|
| Ok | The module abstracted by the agent is working normally |
| NOk | The module has a fault |
| PFO | The module is propagating a fault, that he has generated, through its outbound connections |
| PFOther | The module is affected by a propagating fault on its inbound connections. |
| PFOPFOther | The module is being affected by a fault that is propagating through its inbound connections and that it is propagating over it is outbound connections. |

**Table 1** Internal GSA diagnostic states.



The perceived state of the agent is computed having as base the 18 observations presented in Table 2.

| State | Inbound Minority | Inbound Majority | Sensor Reading | Outbound Minority | Outbound Majority |
|---|---|---|---|---|---|
| Ok | 0 | 0 | 0 | 0 | 0 |
| OMaj | 0 | 0 | 0 | 0 | 1 |
| OMin | 0 | 0 | 0 | 1 | 0 |
| IMaj | 0 | 1 | 0 | 0 | 0 |
| IMaj_OMaj | 0 | 1 | 0 | 0 | 1 |
| IMaj_OMin | 0 | 1 | 0 | 1 | 0 |
| IMin | 1 | 0 | 0 | 0 | 0 |
| IMin_OMaj | 1 | 0 | 0 | 0 | 1 |
| IMin_OMin | 1 | 0 | 0 | 1 | 0 |
| OF | 0 | 0 | 1 | 0 | 0 |
| OF_OMaj | 0 | 0 | 1 | 0 | 1 |
| OF_OMin | 0 | 0 | 1 | 1 | 0 |
| OF_IMaj | 0 | 1 | 1 | 0 | 0 |
| OF_IMaj_OMaj | 0 | 1 | 1 | 0 | 1 |
| OF_IMaj_OMin | 0 | 1 | 1 | 1 | 0 |

| | | | | | |
|---|---|---|---|---|---|
| OF_IMin | 1 | 0 | 1 | 0 | 0 |
| OF_IMin_OMaj | 1 | 0 | 1 | 0 | 1 |
| OF_IMin_OMin | 1 | 0 | 1 | 1 | 0 |

**Table 2** List of possible observations.

Although HMM matrices can be generated from scratch, it is desirable to have a preliminary parameterization representing the adequate behaviour, once the learning mechanism of the HMM is known to be sensitive to the initial parameterization, therefore the HMM parameterization will be further detailed in chapter 4.

The observation perception is held whenever one of the sensors, inbound or outbound readings changes. According to Table 2, and regarding the environment, each GSA might percept one to four realities, inbound minority, if the minority of the inbound interactions is affected by a fault, inbound majority if the majority of the inbound interactions are affected by a fault, the same applies for the outbound, however in this case the interaction involved are the outbound, as the name implies.



Figure 10 depicts two possible environment observations for the central agent. In the A case, the central agent perception is inbound majority, and Outbound minority, on the other hand, in the B case the agent percepts inbound minority, sensor failure reading and outbound majority.

Despite the equality of inbound and outbound interactions, the example just tries to demonstrate how the diagnosable information is perceived in different conditions.


**Figure 10** Possible representation of two surrounding observations for the central agent.

This architecture presents several advantages. Firstly the generic nature and non-domain dependency of the diagnostic system allows its application in most of the network based systems. Furthermore, it favors the emergence of a network perspective through local diagnostic and local information.

Nevertheless this architecture does not exclude or tries to replace the more classical approaches, even because the system just needs information about the condition of the respective device, if it is in failure or not. How the information is processed, is transparent for the developed diagnostic system. Above the sensorial information a classical diagnostic system can be deployed to reason about the internal state of the module. The resultant conclusion of the classical approach would then be returned as the device sensor reading.



### 3.3.3.2 Diagnostic Collaborative Interactions

As was already discussed the diagnostic system rely not only in sensor data, but also in data related to the neighbours agents condition. Whenever an agent enters into a faulty state, it is responsible for alerting its neighbourhood of its faulty condition. This event triggers the diagnostic mechanism that will evaluate the actual state of the GSA according to this new information. Fault propagation is not expected to be synchronous. In this sense the GSA promptly respond to surrounding changes, executing a diagnostic action. If the agent internal state changes into a faulty state, a message is sent to all of its direct neighbours informing of its state, this event triggers a propagation of information in the network. Similarly, when a GSA transits from a faulty state into the normal operation state, it has to report its condition to it's directs neighbours. This behaviour, the diagnostic polarization, favors the comprehension of the

interactions effects, since it allows fault tracing in a stepwise manner. After the recovery of the damaged equipment the internal state will evolve into a normal operation state and the entire network diagnostic will readjust itself to the new conditions.

Figure 11 depicts a fault propagating through the system along with the network diagnostic adaptation and evolution.

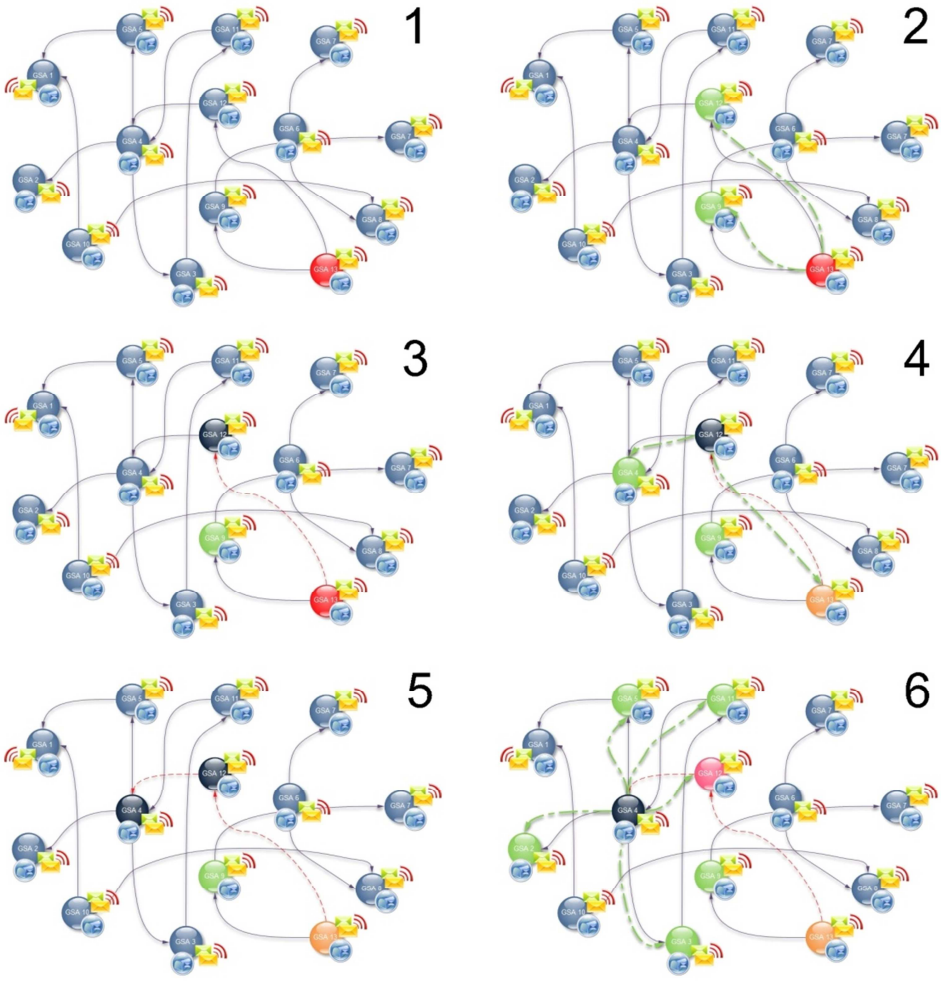




**Figure 11** Local diagnostic adaptation to a fault propagation. Green arrows denote information condition exchange, while red arrows point the fault propagation path.
Color state correspondence – Green: Ok; Red: Nok; Blue: PFOther; Orange: PFO; Pink: PFOPFOther

In the first step an agent inferred that its internal state has changed to a faulty condition. In the second step the agent sent to its direct neighbour information about its internal state. In the third step one of the neighbour agents was affected by the fault and changed its internal state to PFOther, since the fault has been propagated from other agent. The fourth step represents the information of its neighbours about the change of its internal state. Consequently the fault origin agent changes its internal state to PFO. The fifth step depicts the propagation of the fault from the former affected agent. Finally, in the sixth step the last affected agent propagates its faulty internal state, which triggers a change in the internal state of the affected neighbour to PFOPFOther. It is important to stress that the agents only propagate their internal states when it changes from normal to faulty and faulty to normal, independently of the faulty state. At the end a complete network diagnostic is achieved.

Also important to notice, is that with this architecture the absence of a sensor can be suppressed, through the adjustment of the HMM model, giving more preponderant weight to the neighbours information dependents observations over sensorial information. The adjustment can be achieved manually or through local learning.



### 3.3.3.3 Self-Learning

As it was mentioned before, the learning HMM mechanism (Baum-Welch algorithm) depends on initial conditions, once it is a hill-climbing algorithm. Therefore the system is initiated with parameterized matrices according to the prior knowledge acquired by the user/expert. In this sense and since each sensor has a different accuracy, the system can be polarized in order to valorize the information originated by the sensor readings or the information gathered about the neighborhood.

This type of knowledge though, requires historical comprehension of the device or sensor behaviour. In addition there is the need of evolution due to the necessity to cope with the detritions as well with the addition of modules in the system. To cover those points a learning mechanism was implemented.

In [115] some tests and conclusion are presented regarding the learning mechanism. This mechanism was tested in order to verify the impact of repeatability of observations in the performance of the diagnostic system. Baum-Welch is an iterative algorithm, and because, an ideal and consensual number of iterations had to be achieved. The reference was the number of iterations until the stabilization of the B matrix. Using the total number of iterations the system would be overfited, therefore a reasonable value was calculated according to the obtained results. Moreover, the sensibility of the learning mechanism to the number of fault observations used to train the system was also tested.

The implementation of the learning mechanism allows the adaption of the diagnostic system according to the reliability of the information retrieved by both the local sensor and neighbours information. If the sensor loses accuracy through time, the diagnostic system will adapt itself in order to assign more importance to the neighbour information. If the device as a very accurate sensor instead, then the system will learn to trust more in the sensor over the surroundings information.



The observation vectors used by the learning mechanism have to be hand-selected in order to ensure a valid set of observation, otherwise the system could learn inappropriate information.

# 4 IMPLEMENTATION

*In this chapter the implementation of generic manufacturing agents is described. Moreover, the operation of each system component is individually depicted, as well as the test scenarios.*

## 4.1 Multiagent System Implementation

The Multiagent system was implemented over the hardware infrastructure described in chapter 3. Each device is abstracted by a Generic Shop floor Agent. To improve system's scalability, and plug ability, generic code was developed. Therefore, each component shares the same inherent characteristics. The differentiation and instantiation of each GSA is attained by loading an xml file, containing the component specifications and properties, though, every component presented the same GSA inherent competencies.



Regardless the different characteristics acquired by each agent, proper interactions must be established in order to generate the desired infrastructure; thereby, each interaction imposes a notion of neighbourhood between GSA, over which the system may emerge as a whole.

The communication infrastructure, used to exchange information between agents, was implemented through FIPA Request Interaction Protocol. All communications interactions will be further detailed, as well as the operation of each system component.

### 4.1.1 AMI Implementation

As it was briefly described before AMIs (Agent Machine Interface) were implemented with the purpose of bridge the interactions between the GSA and the legacy systems, therefore two types of AMI were created, a Conveyor AMI and a Scara Station AMI. In this sense and since the legacy systems functions were implemented in C++, two dlls (Dynamic-Link Library) were generated in order to enable the call of its functions by the respective AMI.

Aiming to enable offline and online system implementation and tests, two operation modes were created: simulation mode and operation mode.

Both simulation and operation mode have a similar functioning, therefore the same interaction mechanism between GSA and AMI can be used for both models. Instead of calling the dll's functions, a thread sleep is performed when the simulation mode is selected.



In simulation mode, a random fault execution mechanism was implemented in order to simulate real execution faults. A defined threshold between 0 and 1 imposes the percentage of execution faults. For values above the threshold, the execution should occur without any problems, otherwise, it generates a fault. This fault mechanism triggers the fault propagation mechanism implemented in the GSA.

The interactions between GSAs and AMIs were also implemented through FIPA Request Interaction Protocol. Whenever an Agent wants to perform a command, an interaction between the agent and the AMI must be initiated. First a REQUEST message is sent containing fundamental information for the correct execution, this message is then followed by an AGREE message sent by the AMI, confirming the reception of the message. Ultimately, an INFORM message is sent, if it was a valid request, or if the execution went smoothly, otherwise a FAILURE message will take the place of the INFORM message.

The functioning of both AMIs will be detailed bellow.

#### 4.1.1.1 Conveyor AMI

The conveyor AMI is responsible for all the interactions between GSA and conveyors since there is only one controller for all the assembly line (The Flexible Manufacturing and Assembly System - *NOVAFLEX*) conveyors. The conveyors are operated by the AMI through access functions (Table 3) available in the dll.

| Device | Action |
|---|---|
| Conveyor1 …11 | ReleaseFixConveyor1 …11<br>Lift1…11Up |
| Fixture ABB | FixAtABBStation<br>UnFixAtABBStation |
| Fixture Scara | FixAtSCARAStation<br>UnFixAtSCARAStation |

**Table 3** Access functions provided by the Conveyor AMI dll.



Furthermore, the AMI has to be capable of accept simultaneous requests and execute all operations asynchronously, due to the need to instantaneously respond to requests from different GSA, placed on different conveyors of the assembly line. Figure 12 depicts the Conveyor AMI execution process.

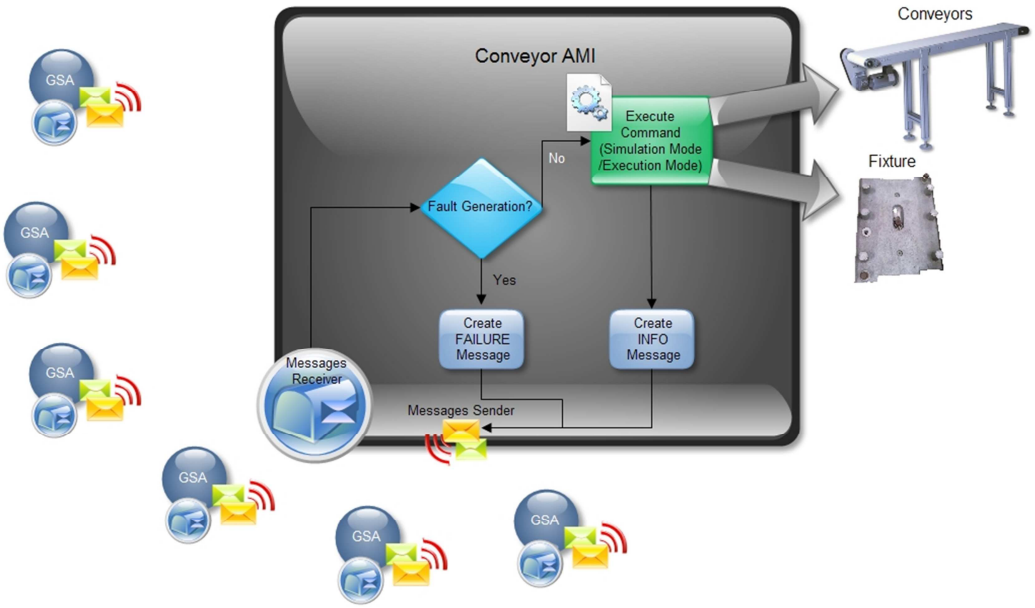


**Figure 12** Conveyor AMI interactions and operation.

If the AMI is running in simulation mode, the command execution is replaced with a thread sleep. Moreover, if a fault is trigered a FAILURE message will be sent otherwise the execution will proceed normally.

### 4.1.1.2 Scara Station AMI

The Scara Station is composed by the Scara robot, and gripper. Consequently the Scara Station AMI is responsible for the integration and operation of both. Unlike Conveyor AMI, Scara Station AMI (Figure 13), has to ensure mutual exclusion in the access of its components by the GSAs. This mechanism unable the concurrent access by multiple agents, which could jeopardize the ongoing operation.



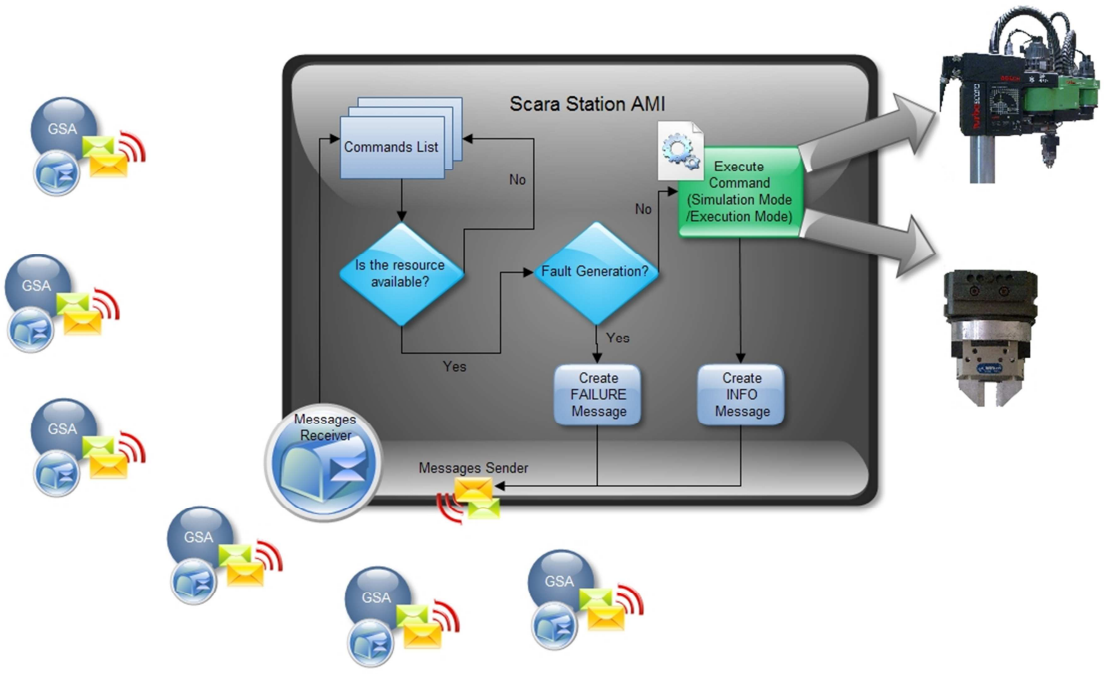


**Figure 13** Scara Station AMI interactions and operation.

To achieve a correct and ordered execution all the requests have to be listed. As soon as the needed resource is available the command will be executed. In the simulation mode the AMI will behave similarly to the previous AMI. The access functions available through the AMI are shown in Table 4.

| Device | Action |
|---|---|
| Robot Scara | MoveNormal |
| | MoveLinear |
| | SetSpeedPercentage |
| Gripper Scara | OpenGripper |
| | UnlockGripper |
| | LockGripper |
| | CloseGripper |

**Table 4** Access functions provided by the Scara Station AMI dll.

### 4.1.2 Broker Agent Implementation

As it was stressed before, the BA only acts as a repository that keeps tracking of all existing composed skills. Whenever a new neighbourhood is generated the BA is contacted by the coalition leader in order to verify if any new skill can be executed by the recently formed neighbourhood. This interaction between the BA and the coalition leader enhances the emergence of skills and consequently introduces new capabilities in the system.



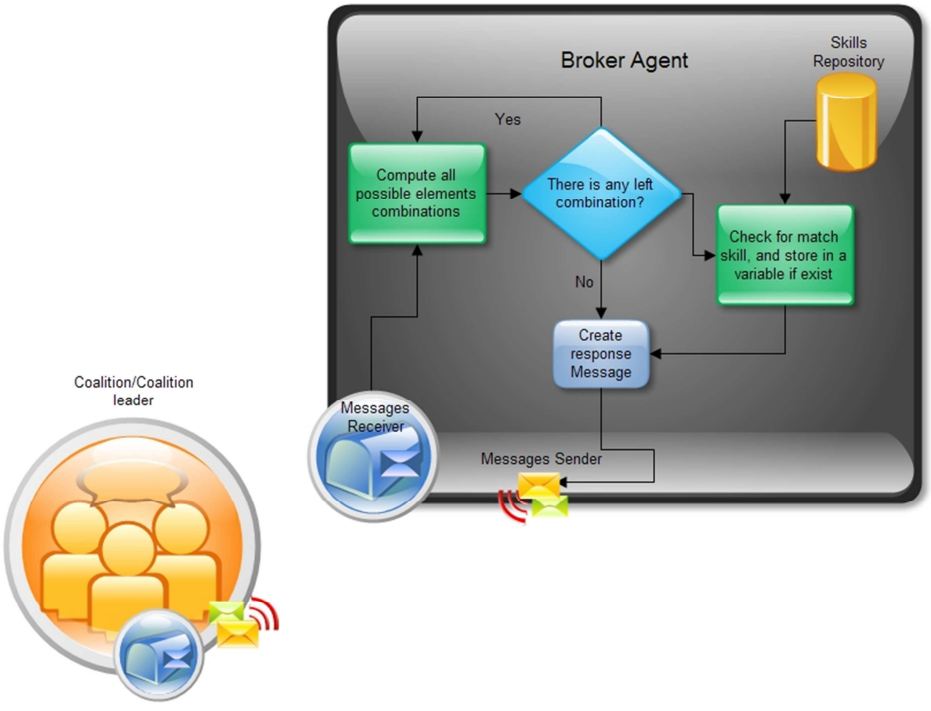


**Figure 14** Broker Agent interactions and operation.

When the BA receives a request from a coalition leader agent, it returns all the skills that can be performed by all elements of the coalition, as well as, the skills that individual or multiple combinations of the belonging elements can perform (Figure 14). Any change in the coalition formation imposes a new coalition leader negotiation between the elements, a contact with the broker, and respective skills actualization.

### 4.1.3 GSA Implementation

The GSA is the main actor in this implementation prototype. As stressed before, the GSA is a generic agent which can acquire the specifications of any system intervenient device.

#### 4.1.3.1 Instantiation Mechanism

Since the GSA is a generic agent, the first step is the agent instantiation in order to differentiate each GSA in the system, according to its function. The agent native skills, as well as the preliminary setup specifications, are loaded from a pre-defined xml file (Appendix 1). The file is created by a human expert who specifies the agent name, type, reference, native skills and AMI through which the GSA will execute the actions, among other information. When created, the agent starts loading the xml file to a string, from which is parsed to Skill structure, Table 5.



The agent instantiation consists on setting the agent parameters according to the ones defined in the xml file.

| Skills Class | |
|---|---|
| AgentName | Contains the agent name |
| AgentType | Contains the agent type |
| Agent Reference | Variable where any particular reference of the agent can be defined |
| WorkflowManagerName | Defines the actual workflow manager name |
| WorkflowManagerType | Defines the actual workflow manager type |
| SkillList | List of skills that the agent can perform |

| | | | |
|---|---|---|---|
| | Skill | Composed Skill | Is defined by a name, AID of the agent that will perform the skill and a unique id that identifies the skill. A set of arguments can be set up to allow a possible instantiation of the atomic skills. The nature and the required modulo to execute the skill is also defined by the composed skill. The CompositionGraph is where a set of atomic skills can be defined. With a composed skill an higher level skills can be instantiated |
| | | Atomic Skill | The name of the atomic skill sets the operation that the skill will execute. Is also defined by a unique id, an AID that will contain the agent AID which will perform the skill. As a set of atomic skills can form a higher level skill, the id of the previous skills that need to be executed is also defined. The type of the model that can execute that atomic skill is also defined. |
| | | Restrictions | |
| | | AlternativeSkill | Variable containing the id of the alternative skill to execute in case of a failure. |
| | | isInstantitated | Variable where is defined if the skills is or is not instantiated. |
| | | Nature | Defines the skill type, if it is Composed or Simple Skill. |
| | | RequiredModules | Variable where is specified the type of agent that can execute the respective skill. |
| AlternativeSkillList | List of skills that contain the alternative skills that an agent can perform, as an alternative to a failed skill execution. Each skill can be an Atomic Skill or a Composed Skill. | | |
| AMItype | Variable where can be defined the agent correspondent AMI type. | | |
| AMIName | Variable where can be defined the agent correspondent AMI name. | | |
| Restrictions | Defines the restrictions that are needed to take into account when both, composed skill or atomic skill, are executed. In each one of these sub-classes a set of restrictions can be defined | | |
| WorkflowManagerPosition | Provides the real time location of the workflow manager | | |
| RawData | Variable that contains any extra necessary information | | |

**Table 5** Structure of a Skill Class



### 4.1.3.2 Neighbourhood Establishment

The connections between agents are defined through a neighbourhood system. Moreover, the system infrastructure is set as the neighbour interactions are established.

All negotiation neighbour process is facilitated by a neighbour class, Table 6. This class allows a easier and more intuitive negotiation process since it defines the information needed and exchanged when a negotiation is performed. The neighbour class contains parameters like neighbour information and status, as well as a list of interactions and a list of skills. To represent an interaction, an Interaction class was also created. Through the Interaction class it is possible to define the interaction properties and all the necessary state and operation information. Whenever a neighbour negotiation takes place, a neighbour object is created and used to exchange information between the intervenient agents.

| Neighbour Class | | |
|---|---|---|
| Interaction List | Sense | Variable where the interaction sense is defined. It may assume one of the following values INBOUND, OUTBOUND or BOTH. |
| | Type | Defines the interaction type which can be PNEUMATIC, HYDRAULIC, ELECTRICAL, MECHANICAL, FLUX or COMUNICATION. |
| | State | Variable where the interaction state, is defined. |
| | Operation | Variable where the operation to execute is defined. The operation can be ADD, REMOVE or UPDATE. |
| | Counter | Auxiliary variable to control the interactions number. |
| Skill List | List containing the skills ids that the agent possesses. | |
| Neighbour AID | Contains the requester agent AID. | |
| Neighbour Type | Contains the requester agent type. | |
| Neighbour Reference | Contains the requester agent reference. | |
| Neighbour Name | Contains the requester agent Name. | |
| Status | Specifies the neighbourhood status between the intervenient agents (CONFIRMED, REQUESTED, COMMITED, UNDEF, REFUSED, FAILURE, PENDING_INTERACTIONS). | |
| Operation | Variable where the operation to execute is defined. The operation can be ADD, REMOVE or UPDATE. | |

**Table 6** Structure and content of the Neighbour Class.

To initiate and establish a neighbour relation, the following message sequence must be satisfied, Figure 15.

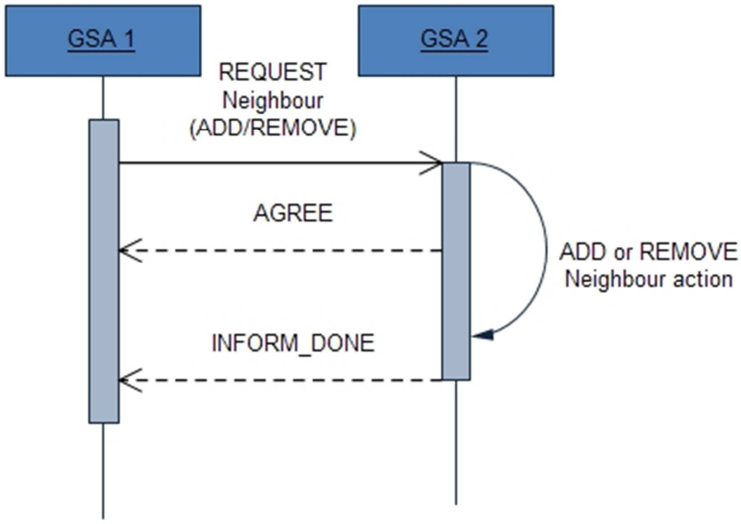


**Figure 15** Interactions to ADD or REMOVE a Neighbour.

The requester starts sending a message to the agent with whom it intends to establish a neighbour relation. The receiver returns an agree message and starts processing the request message. If the message is processed without any problem an inform message is then sent to the operation requester and the operation terminates. Otherwise, if any problem occurs, a failure message is sent instead. In the requester case, when the agree message is received; a similar operation to the one performed by the receiver, takes place. If the next message received is an inform, the operation is confirmed and the neighbour relation was successfully established. If, on the contrary, the received message is a refuse, then the request takes no effect.



When a neighbour relation is correctly established, both agents have the other agent in the neighbours list, eventually, only with a different interaction sense. In Figure 16 a representation of the neighbourhood interactions of NOVAFLEX is depicted.

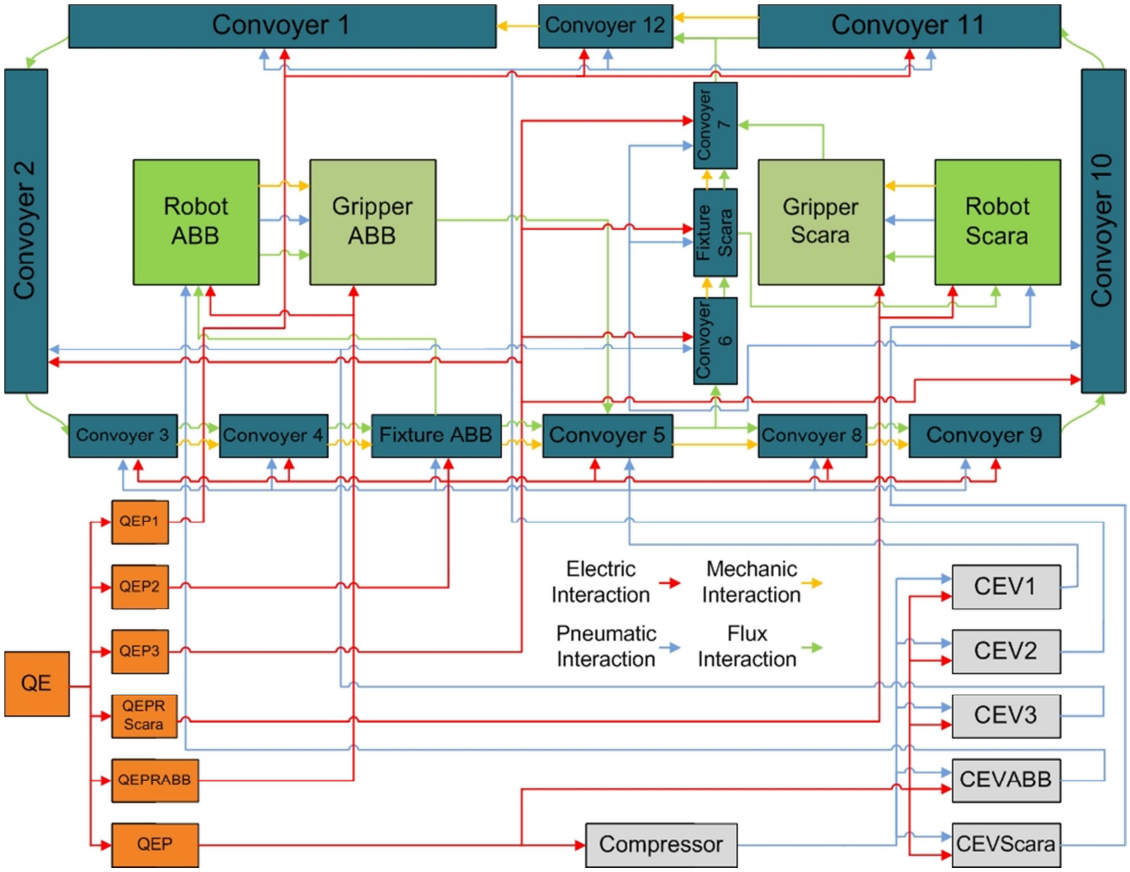


**Figure 16** Representation of a possible neighbourhood scheme for the NOVAFLEX cell.



## 4.1.3.3 Broker Interaction

A broker Interaction is performed whenever a neighbour interaction is established and the agent assumes the coalition leader roll. The definition of the coalition leader is achieved through a function that verifies the agent hierarchy. If the agent is the highest hierarchical intervenient then a Broker Agent request is performed. In order to successfully establish a correct interaction with the broker, the following message sequence must be respected, Figure 17.

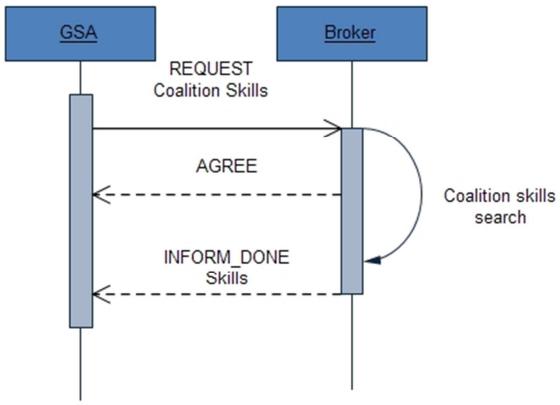


**Figure 17** Messages exchanged between GSA and the Broker Agent.

The request message sent by the GSA to the Broker Agent contains the information about all the agent neighbours, type and reference Table 7, which will allow the Broker Agent to search for the skills that can be performed by the different combinations of all the constituent agents. As response, and after the Skills search, the Broker Agent generates and sends to the GSA an INFORM message containing a Skills object with all the coalition executable skills. Finally the GSA skills are updated in order to accommodate the new skills. The process is repeated whenever a coalition agent is added or removed.



| Agent Type | Message | Information | | |
|---|---|---|---|---|
| GSA | Broker request message | Neighbours Agents Information List | Type | Contains the agent neighbour agent type |
| | | | Reference | Contains the agent neighbour agent reference |
| Broker | Broker response message | Skills | | |

**Table 7** Information exchanged when a broker request is performed.

### 4.1.3.4 Skill Instantiation

In order to present emergent skill behaviour, each GSA has to be aware of its own skills. However, the information stored consists only of the skill skeleton. To execute a skill, the GSA as to instantiate the requested skill with the received operation request information. The instantiation process consist on the completing the skill skeleton structure with the needed information of the operation.

The execution of an operation can be requested through a composed or a simple skill depending on the type of operation. In the composed skill case, the skill may have to be executed by a number of agents, since the skill is composed by a set of simple skills that may not be executed by the same agent. Moreover, the agent responsible for the skill execution is also responsible for its instantiation. When the agent receives the skill, the agent starts verifying if the skill is instantiated. If so, the agent proceeds to the respective skill execution, otherwise the skill instantiation is performed.

For the simple skill the process is similar, however, only instantiation verification is performed. When an agent is asked to execute a simple skill, it is considered that it can execute the skill, otherwise a failure will occur and the error recovery will take place.

The skill instantiation process can be divided in three major phases, Figure 18, the skill confirmation, the arguments instantiation and the executer instantiation. The skill confirmation verifies if the skill is one of the agent native skills. If the result verification is positive then the skill will be instantiated. The argument instantiation ensures the correct attribution of all the parameters needed to the execution of the skill. Finally, the executer instantiation varies according to the existence or not of restrictions. A proper executer, with all the necessary characteristics, is selected. If the actual agent presents all the required characteristics it will be the preferential executer.

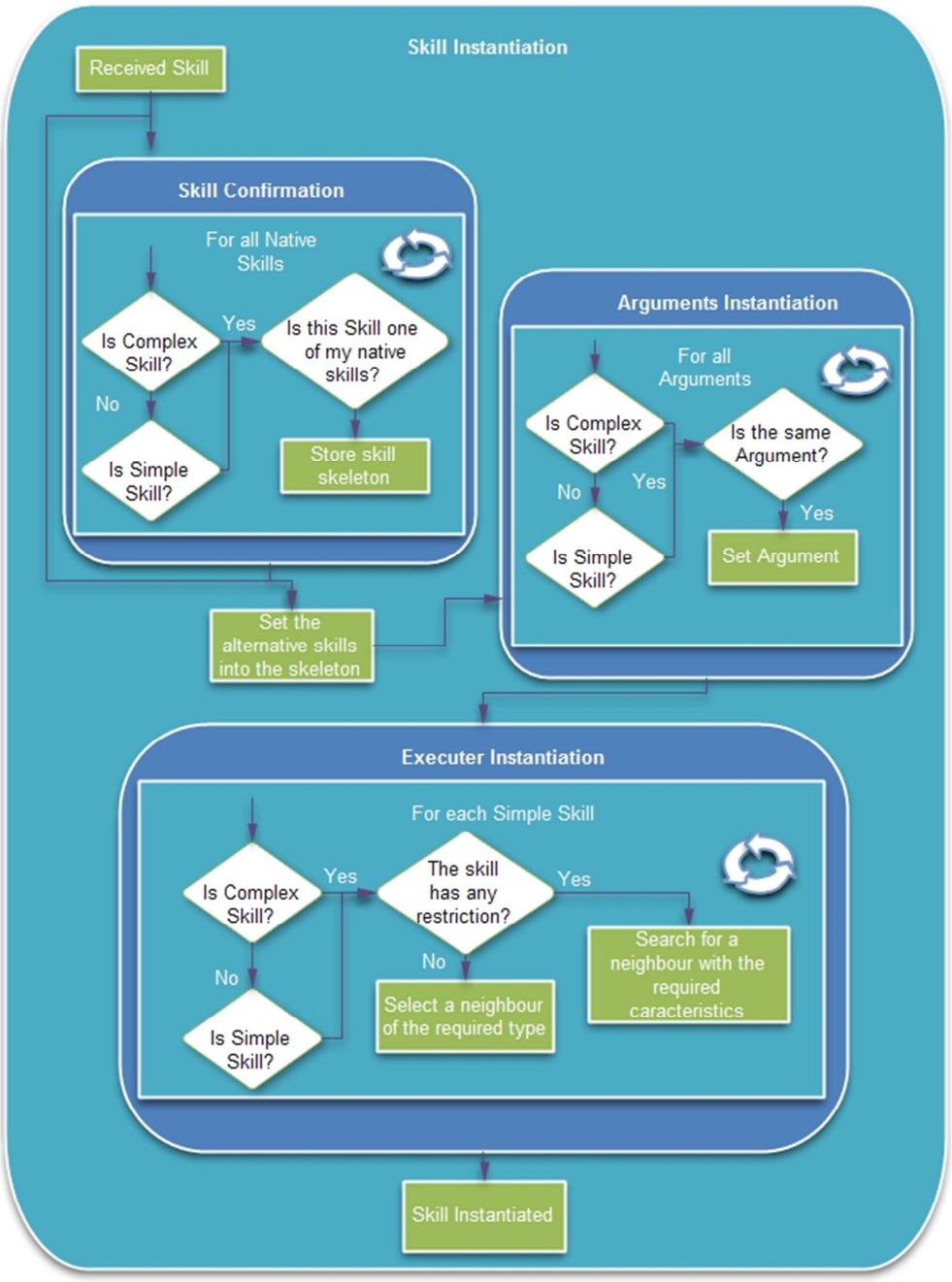


**Figure 18** Skill instantiation flowchart.

### 4.1.3.5 Orchestrator

The orchestrator is the module responsible for the skill execution management (Figure 19). Whenever a GSA receives a composed skill execution request an orchestrator is launched. This allows parallel management and execution of more than one skill at the same time.

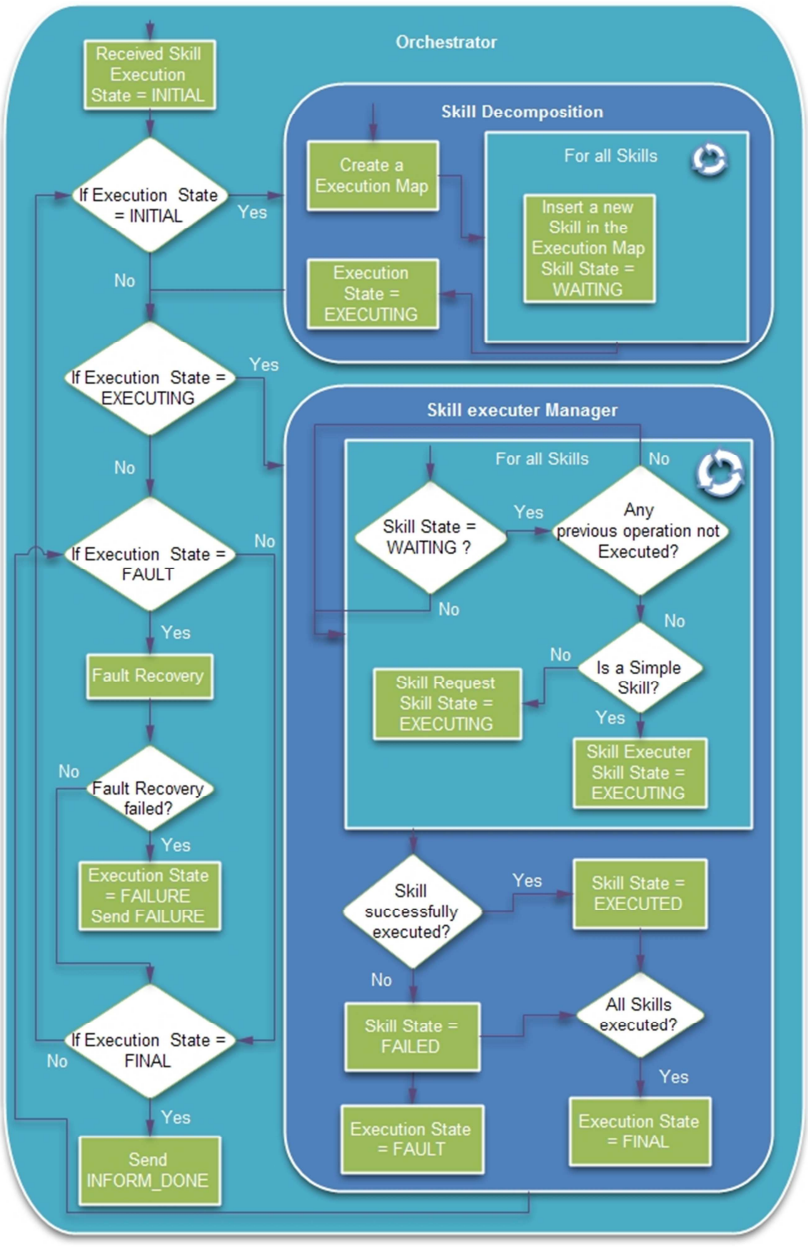


**Figure 19** Orchestrator flowchart.

To better understand the figure description it will be considered that the received skill is a composed skill, while the skills composing the received skill will be named as skill.

Firstly, the execution state is settled as INITIAL. Moreover, an execution map is created containing all the skills that compose the composed skill. To each skill is attributed the state WAITING.

The execution order of each skill is defined through precedence's system. Before the execution of a skill the precedence variable is checked, in order to verify which skills must be executed first. If all the required skills have already been executed, then the execution of the actual skill can proceed and the skill state is set as EXECUTING. The skill execution varies, according to the skill type. For the simple skill case, a skill executer is performed, otherwise a skill request is launched instead, which will generate, in the receiver GSA, a new orchestrator to deal with the skill. After the skill execution, the skill state is updated to EXECUTED if everything occurred as planned. On the other hand, if something goes wrong the skill state is set as FAILED. A verification is then performed, in order to verify if all the skills have been performed, thereby in that case the execution state is updated to FINAL. If any skill failed, the execution state becomes FAILED.



If the execution state variable assumed the FAILED value then fault recovery takes place. Otherwise the execution state presents the value FINAL and the execution is terminated by sending an INFORM_DONE message. The same process is performed if the fault recovery process has been successfully executed. If not, a FAILURE message is sent instead.

Since the orchestrator is implemented through a behaviour, all processes will be repeatedly executed until the sending of one of the INFORM_DONE or FAILURE message.

As an example Figure 20 presents the correct message exchanged between the intervenient actors to achieve a correct pick and place skill execution.

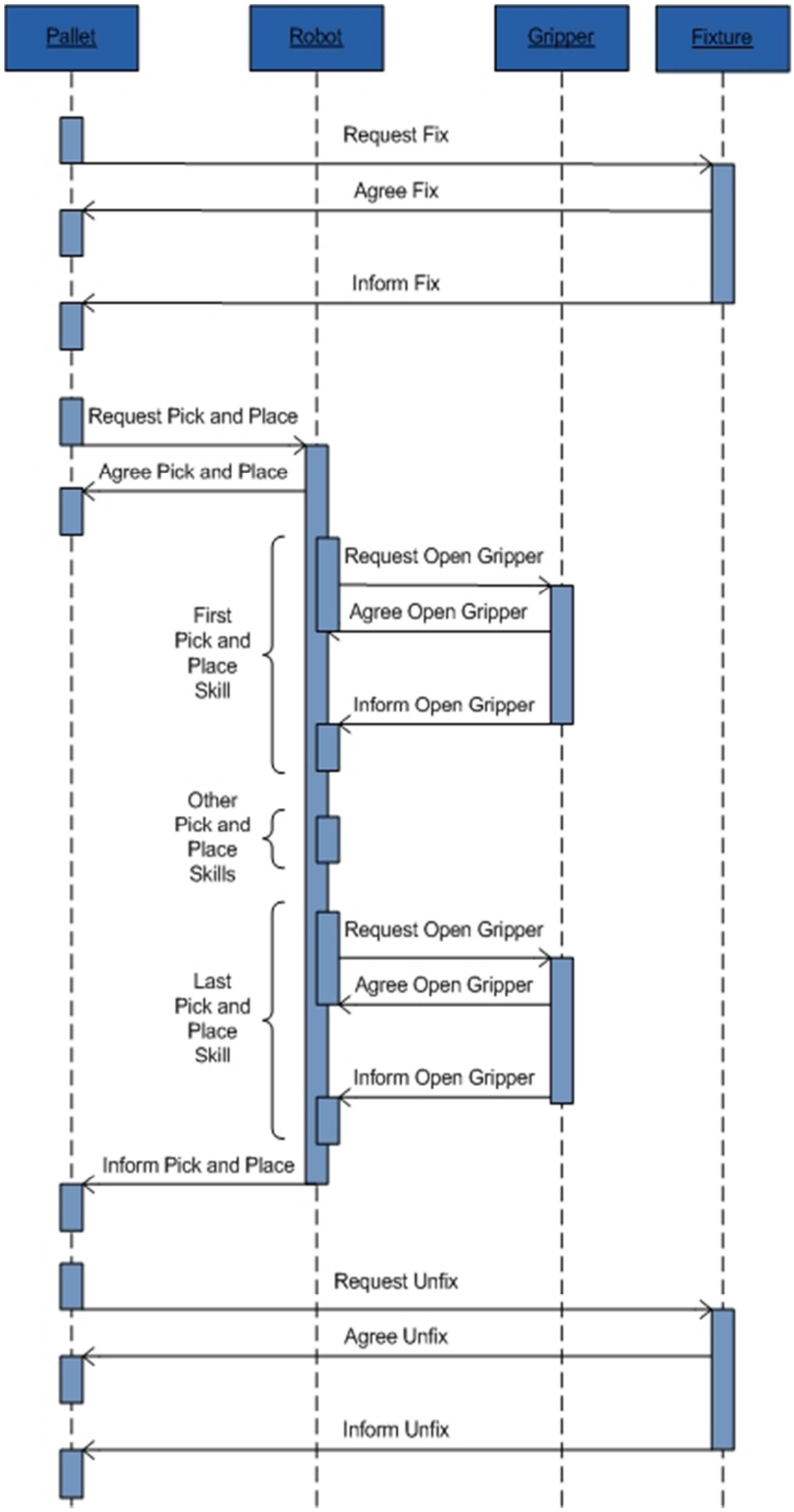


**Figure 20** Abbreviated message in a Pick and Place operation.

### 4.1.3.6 Fault Recovery

The fault recovery process is responsible for the instantiation of the failed executed skill with another executer agent or, if no other is available, for the execution of alternative skills as an attempt to overcome the failed execution of a skill. Each skill can be associated with a defined alternative skill that can be used in recovery actions. The recovery process, Figure 21, is initiated by the orchestrator, whenever the execution state is FAULT.

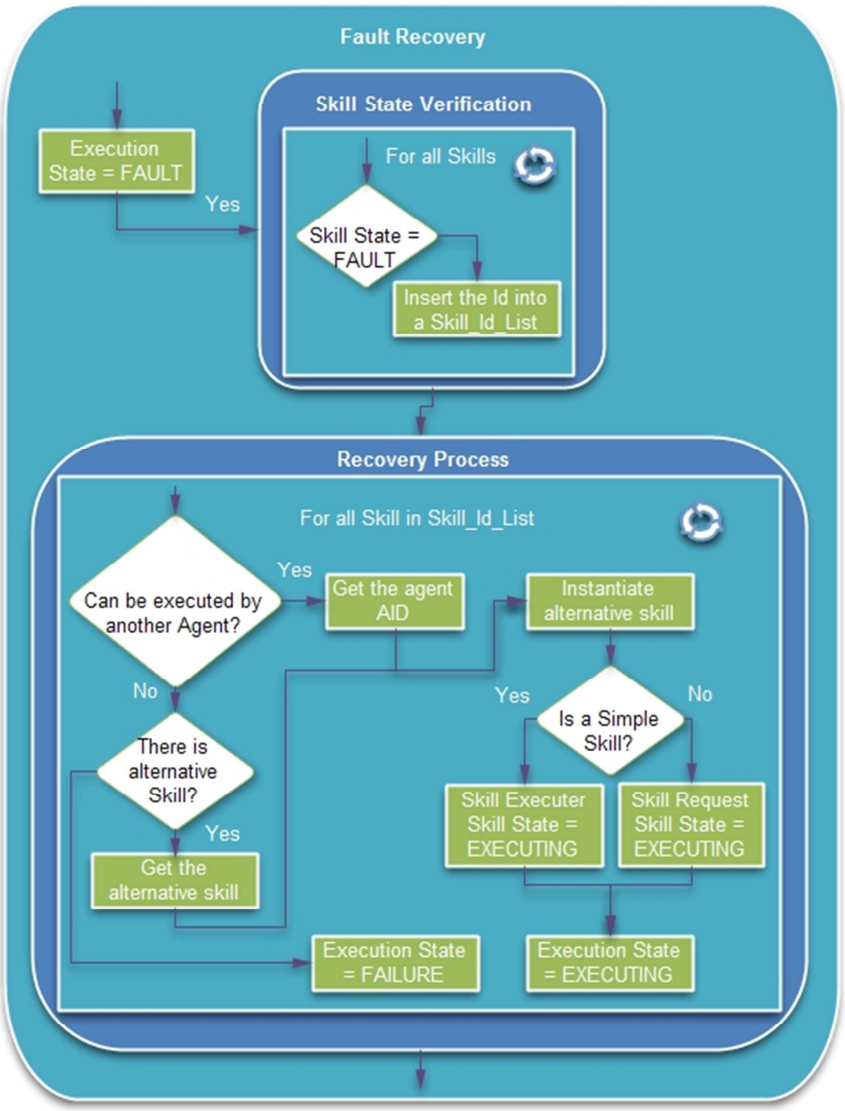


**Figure 21** Fault Recovery flowchart.

The process starts listing the FAILED skills into a list. For each skill in the list a recovery process will be executed. The recovery process starts verifying if there is any neighbour agent available to execute the skill. If any agent is available then the Skill will be re-instantiated and executed according to its nature. On the other hand, if no agents are available to proceed with the execution, the skill is checked in order to verify the alternative skill, which if successfully executed will allow the correct execution of the composed skill. In case the alternative skill has not been correctly executed, the same process will be repeated until there are no more alternative skills. If no alternative skill was successfully executed then the execution of the composed skill fails and the execution state is set as FAILURE, otherwise is set as EXECUTING.

It is important to refer that the fault recovery will not solve the fault it simply contours the problem executing another skill.

#### 4.1.3.7 Workflow Manager



In this work, a workflow is a series of action that can be executed simultaneously and sequentially. The module responsible for the correct and ordered execution of the pretended workflows is the workflow manager.

A workflow is defined by the user/expert through a file containing the xml structure of a Skills object. As shown before, a skill object can define a SkillList, moreover each ComposedSkill can be constituted by a number of SimpleSkills. In this way a workflow can be thought as a skill which is defined by a set of skills.

Each GSA, independently of the abstracting entity, is able to manage workflows. However, this role is usually played by the abstracted entity that will be processed. Therefore, the workflow manager has to be able to move over the shop floor by itself or through another entity. The workflow manager can only request to its direct neighbours the execution of skills. Because of this constraint, the workflow manager neighbourhood cannot be static, but adjustable according to the workflow position in the shop floor. In order to execute the skill requests, the workflow manager has to establish interaction between GSAs. Since not all GSA abstract the same entity type, different interactions, defined in the restriction variable of the Skills class, are

established according to its type. Whenever an executed skill implies a change in the workflow manager position, its neighbourhood is updated and has to match the local entity neighbourhood.

Figure 22 depicts the GSA workflow manager neighbourhood evolution, when the transition from position 1, to position 2 takes place through the execution of a move skill.

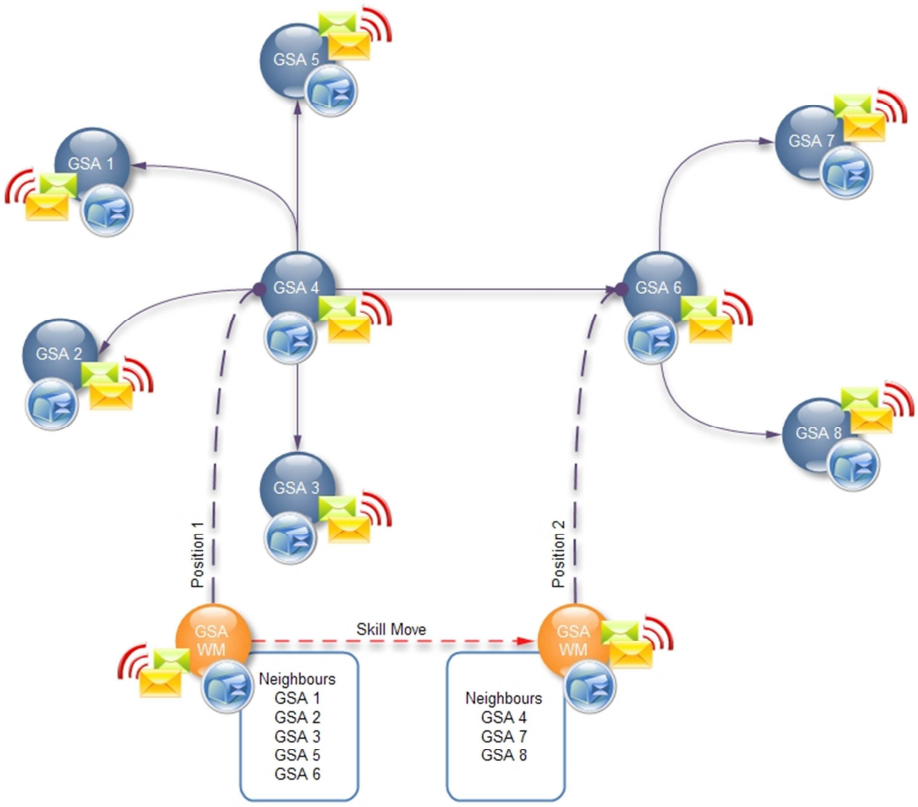


**Figure 22** Workflow Manager Neighbourhood before and after the execution of a move Skill.

The message sequence respecting the change from position 1 to position 2 by the workflow manager is depicted in Figure 23. It is important to underline that when the position change takes place the workflow manager GSA neighbourhood is removed before the neighbour's list request. This action simplifies the neighbourhood management process in the workflow manager.

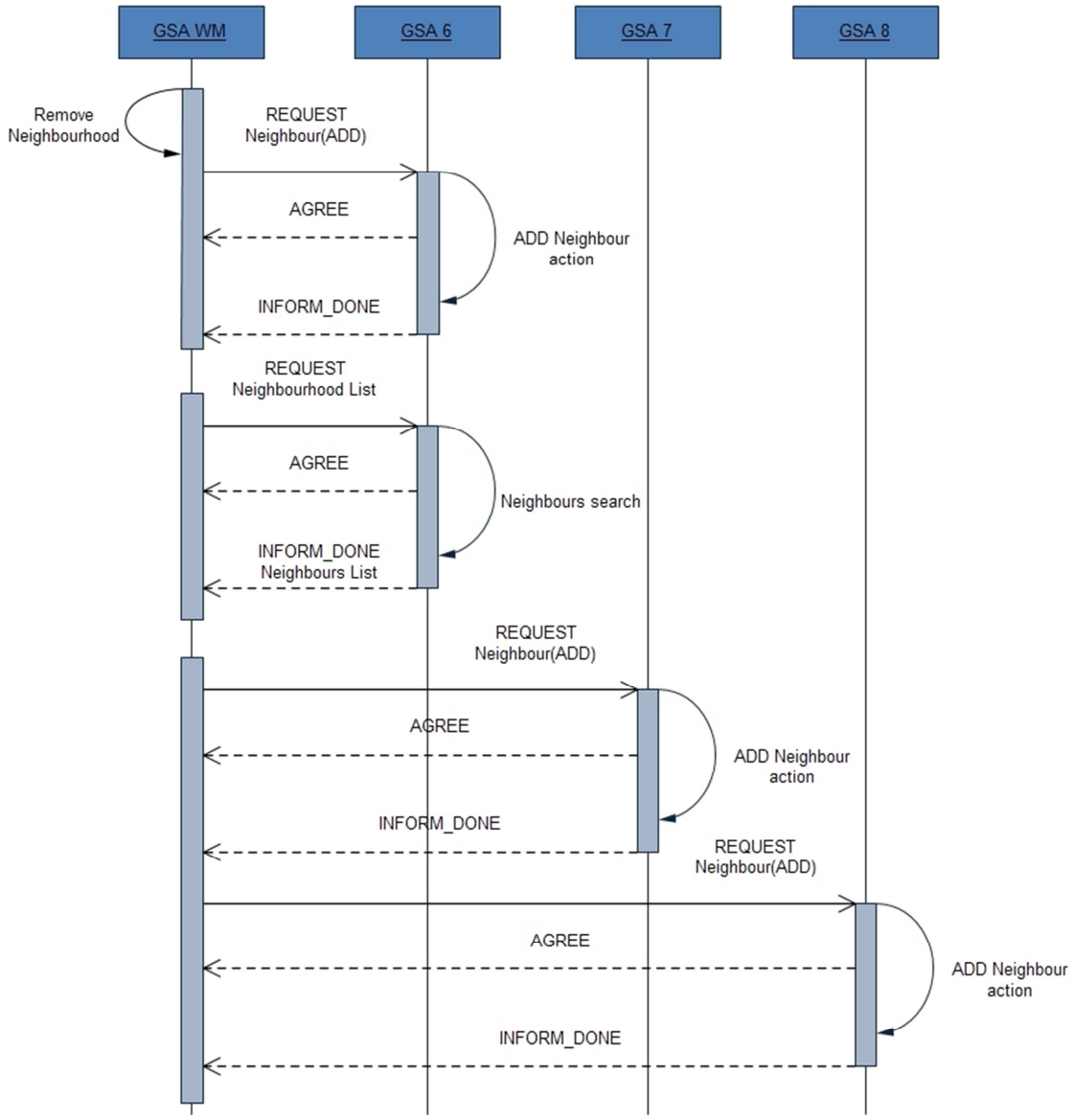




**Figure 23** Abbreviated message exchanged during the change from position 1 to position 2.

The workflow manager process, Figure 24, starts with the load of the workflow xml file. After the load and parsing of the file to a Skills object the execution state is set as INITIAL. Similarly to the orchestrator process an execution map is created, which will be used to control the skills execution. The skill state of each skill is set as WAITING. As stated above, whenever the workflow manager has to change position new interactions are established according to the

restrictions imposed by the workflow. To facilitate the process execution, a restriction map is created.

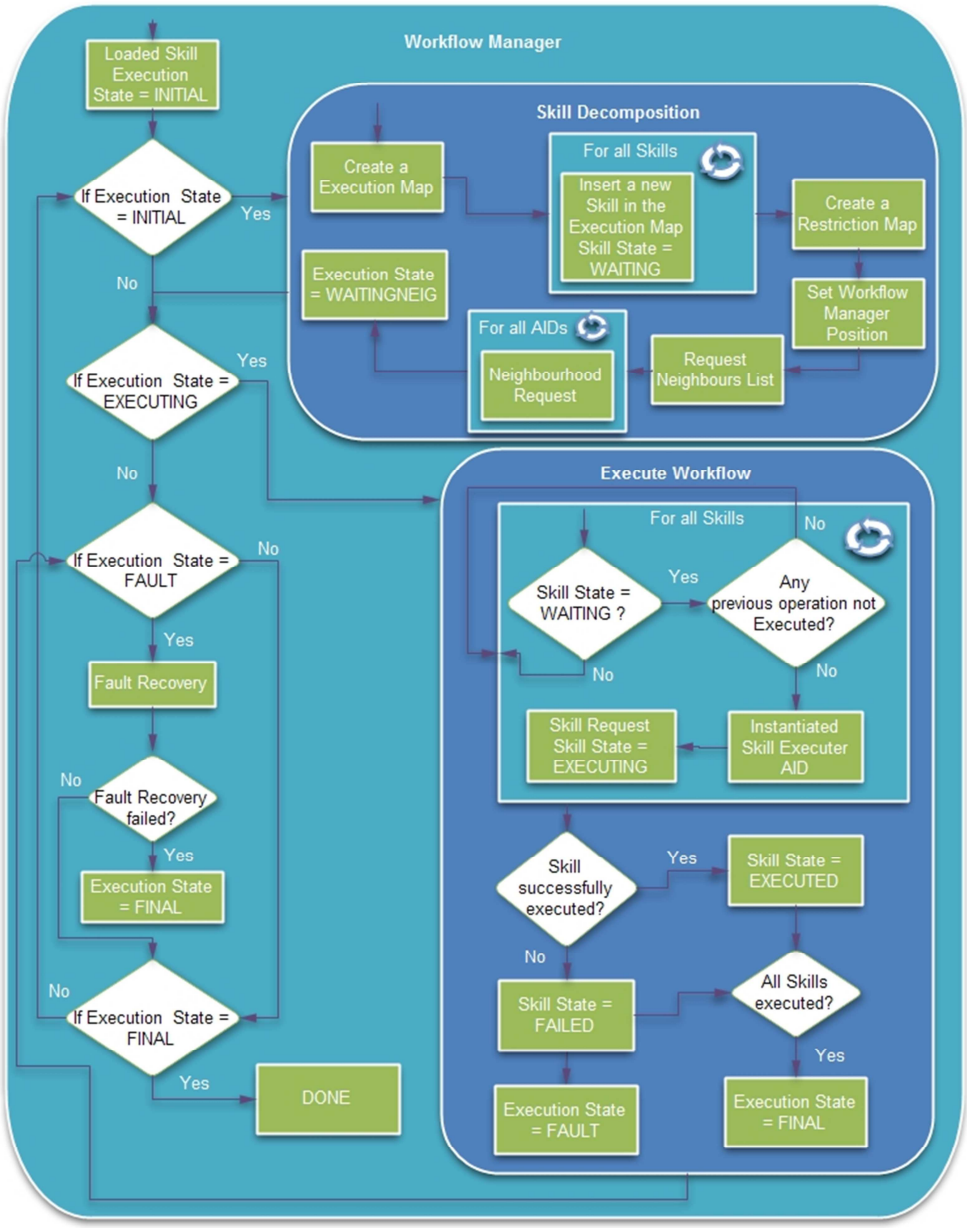


**Figure 24** Workflow Manager flowchart.

The initial position of the workflow manager GSA is also defined within the workflow file and indicates the entity with which the workflow manager GSA must establish its first

interaction. After the interaction establishment a request is sent in order to obtain the agent neighbourhood list which will be used to make the neighbourhood requests. After the submission of all the requests the execution state is defined as WAITINGNEIG. The process will only be resumed when all the neighbour confirmations are received and the execution state changes to EXECUTING. As soon as the execution state is set as EXECUTING the workflow execution will start.

The workflow execution is at all similar to the execution of a composed skill by the orchestrator, only with a few small differences. The skill execution is always performed by another GSA, whereas in the orchestrator a skill can also be executed by the same agent. The workflow manager, such as the orchestrator, has the fault recover capacity. However, in this case when the fault recovery fails the execution is set as FINAL, which implies that the workflow was terminated without having been fully executed.



### 4.1.3.8 Fault Propagation Manager

A GSA network (Figure 25) is a collection of connected nodes, were each node is a GSA abstracted entity. In manufacture, each one of these entities will be abstracting real shop floor equipment, respective actuators and sensors, from where the information is collected during the execution. Nevertheless, a GSA network allows fault simulations which are controlled by the fault

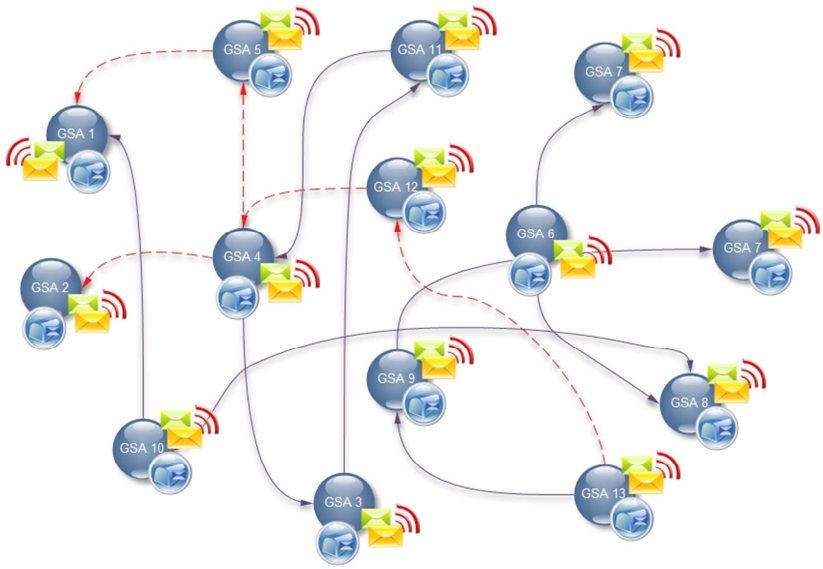


**Figure 25** Example of a GSA network Fault propagation.

propagation manager in each GSA.

Three propagation methods were implemented the random propagation, random vulnerability and GSA type sensitive method. The random propagation method implies equal probability propagation for all types of GSAs. The random vulnerability method uses the random propagation method. However the fault acceptance probability is defined according to the condition of the GSA. If the GSA is randomly selected to be a vulnerable node then a high fault acceptance probability will be attributed, otherwise a it will be attributed a low fault acceptance probability. The GSA type sensitive propagation methods, consists on a higher propagation

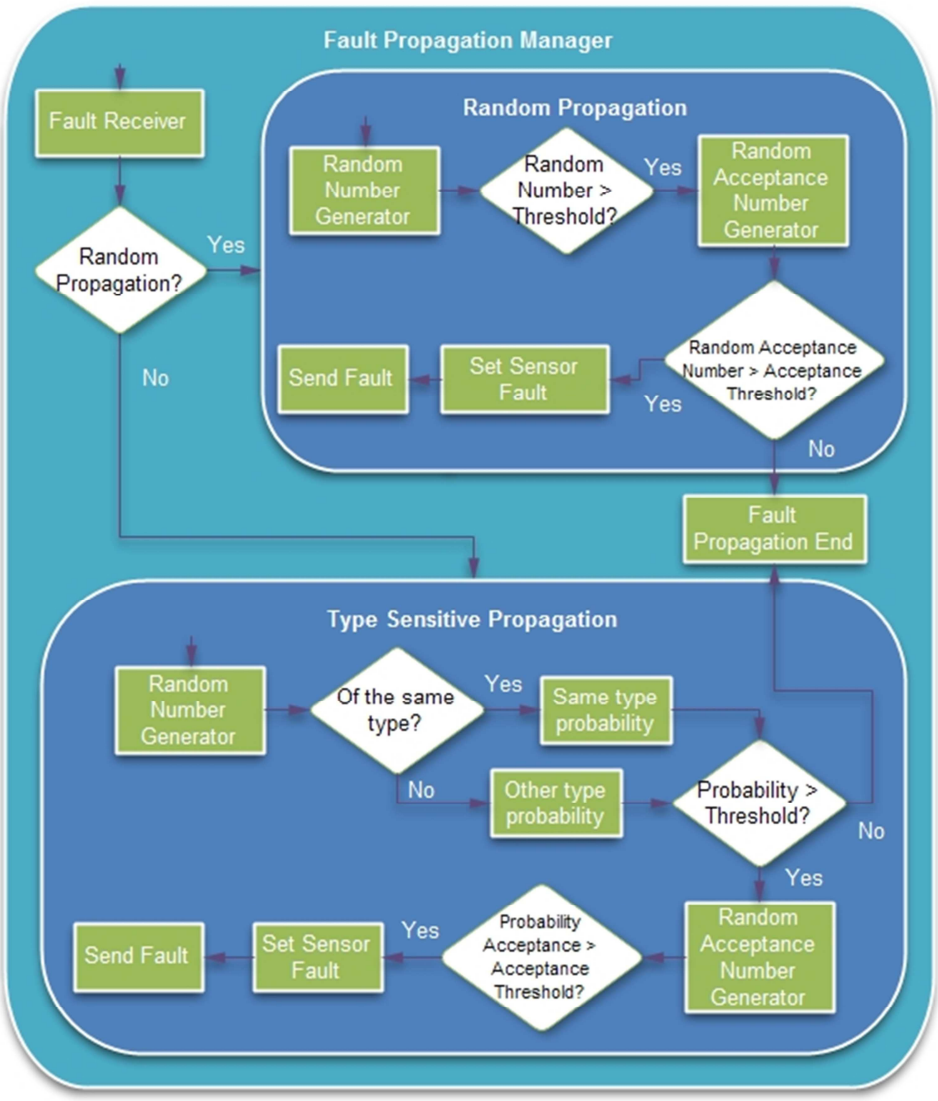


**Figure 26** Fault Propagation Manager Flowchart.

probability for the same GSA type. Figure 26 describes the fault propagation manager execution.

If the fault propagation option is activated, the manager is executed whenever a fault request is received. The receiver starts the execution checking the propagation method activated.

All propagation methods are similar. The major difference is that for random propagation the probability is always the same, while for type sensitive random propagation the agent receiver type is verified and the probability is selected accordingly. The random vulnerability method assumes the probability according to the node condition (vulnerable or not).

After de method verification a random number is generated and compared with the user/expert defined threshold. Although rarely, sensors can present misleading readings, another threshold was established to simulate that behaviour. After the generation of a new random number both values are compared, if the value of the sensor is greater than threshold, then the sensor is not working properly and a fault is assumed by the agent. Otherwise, the fault will not affect the agent but it will be equally propagated.



It is important to stress that whenever a fault is accepted, if the fault propagation simulation is activated and even if the sensor is damaged, the fault will always be propagated. It is the receptor that will decide if accepts the fault or not.

In order to represent the fault propagation in the diagnostic interface, a communication system was implemented. Whenever a fault is propagated a message is sent to the GDDA informing the agent sender, receptor, and the interaction type. The interaction is established through a simple FIPA REQUEST protocol, Figure 27.

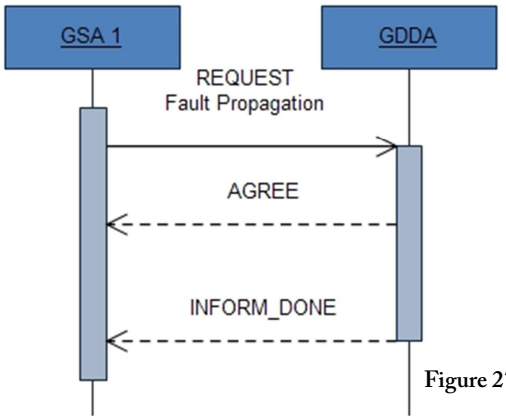


**Figure 27** Message exchanged between GSA and the graph designer.

### 4.1.3.9 GSA Interface

A GSA is represented through an interface which establishes the connection between each agent and the system user/expert. Through the interface, Figure 28, the user/expert can

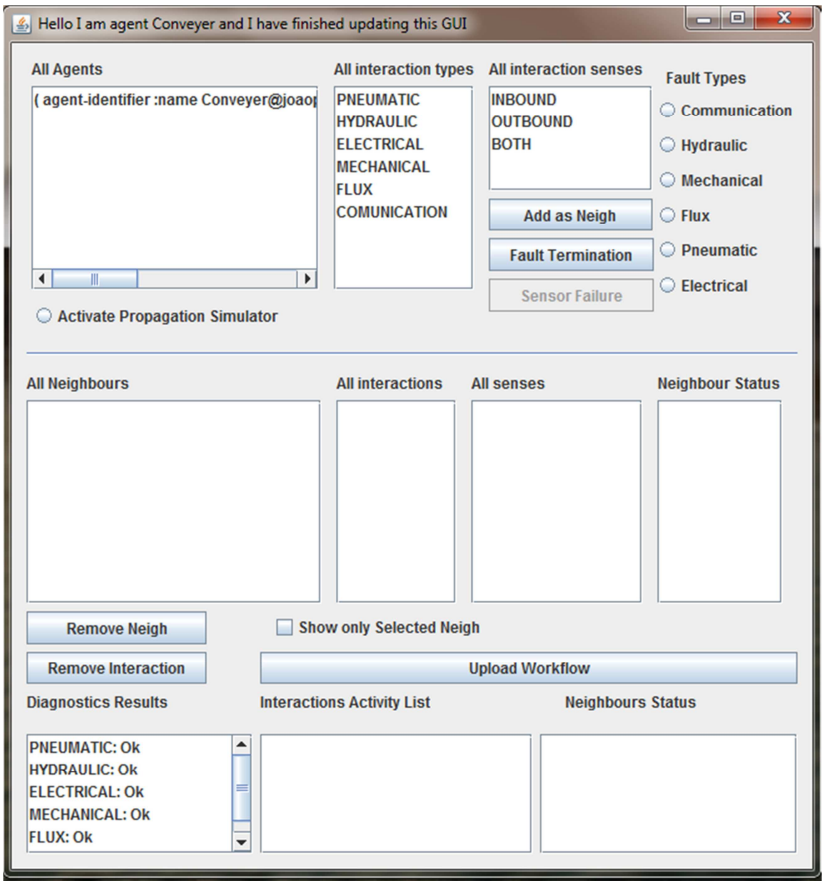


**Figure 28** GSA interface.

manage the agent state using the information displays, as well as manage the agent neighbourhood. It can also manually insert a local fault, with fault propagation or not, and load a workflow. When the learning mechanism is activated the following interface appears, Figure 29.

The fault observations are presented in the display. When selected the fraction of learning interactions a button will be available in order to execute the learning algorithm, otherwise the user/expert can always press the No button and cancel the learning action.

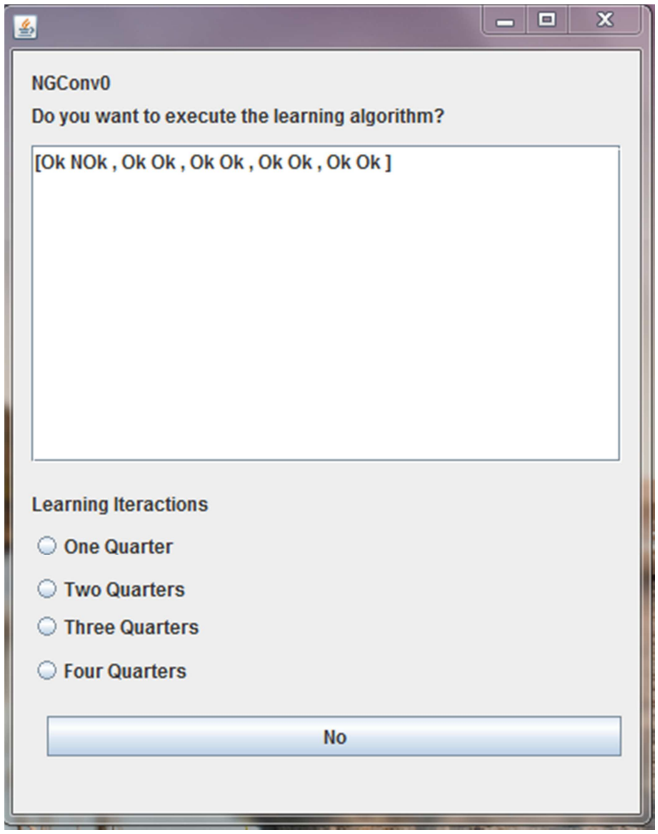




**Figure 29** Learning algorithm interface.

### 4.1.4 GSA Diagnosis System

As it was stressed before, the diagnostic system is based on agent sensors and neighbours information. In order to acquire neighbours state information, a diagnostic communication system was implemented.

Since a distributed adaptive diagnostic is the goal, each agent has to be aware of its surroundings, which is achieved through exchange of status information. There are only two conditions that trigger the sending of status messages, the transition of agent status from "OK" to any other state and from any other state to "OK". Figure 30 shows the status propagation, trace arrows, from GSA 4 to all of its direct neighbours, due to a transition from "OK" state to a faulty state. The receiving agents will then know that one of its direct neighbours, inbound or outbound, has experienced a failure and therefore they will also diagnose themselves in order to update their inner status. In case their status has also changed to a faulty state the process will be

repeated, otherwise nothing will happen. This process allows the correct exchange of status information between neighbours.

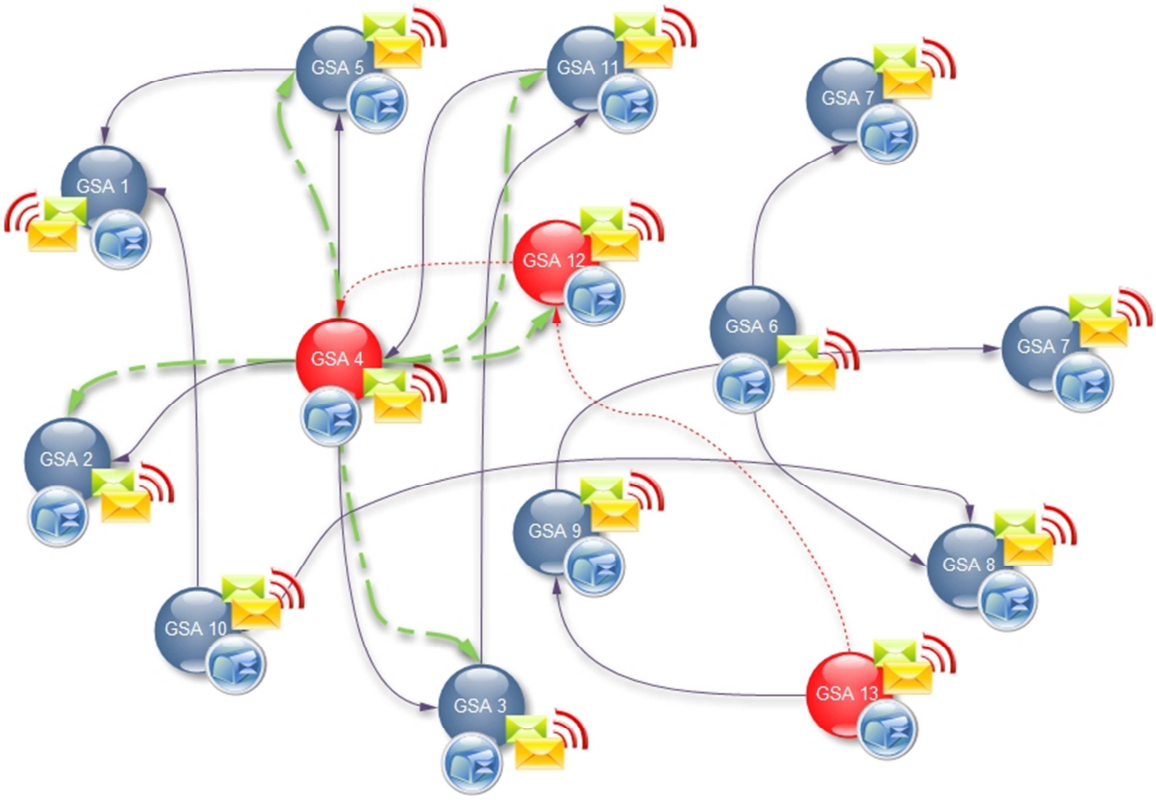


**Figure 30** Example of a GSA network Status propagation (trace arrows).

The necessary message sequence to inform the neighbours of the change in an agent status is shown in Figure 31. After the diagnostic execution and subsequent change of state, due to the fault reception, the agent informs its neighbours of its faulty state sending a message with its status. If, on the contrary, the sequence is not verified, then something went wrong and the fault print in the network will be erroneous, which implies that all the conclusions will not be valid.

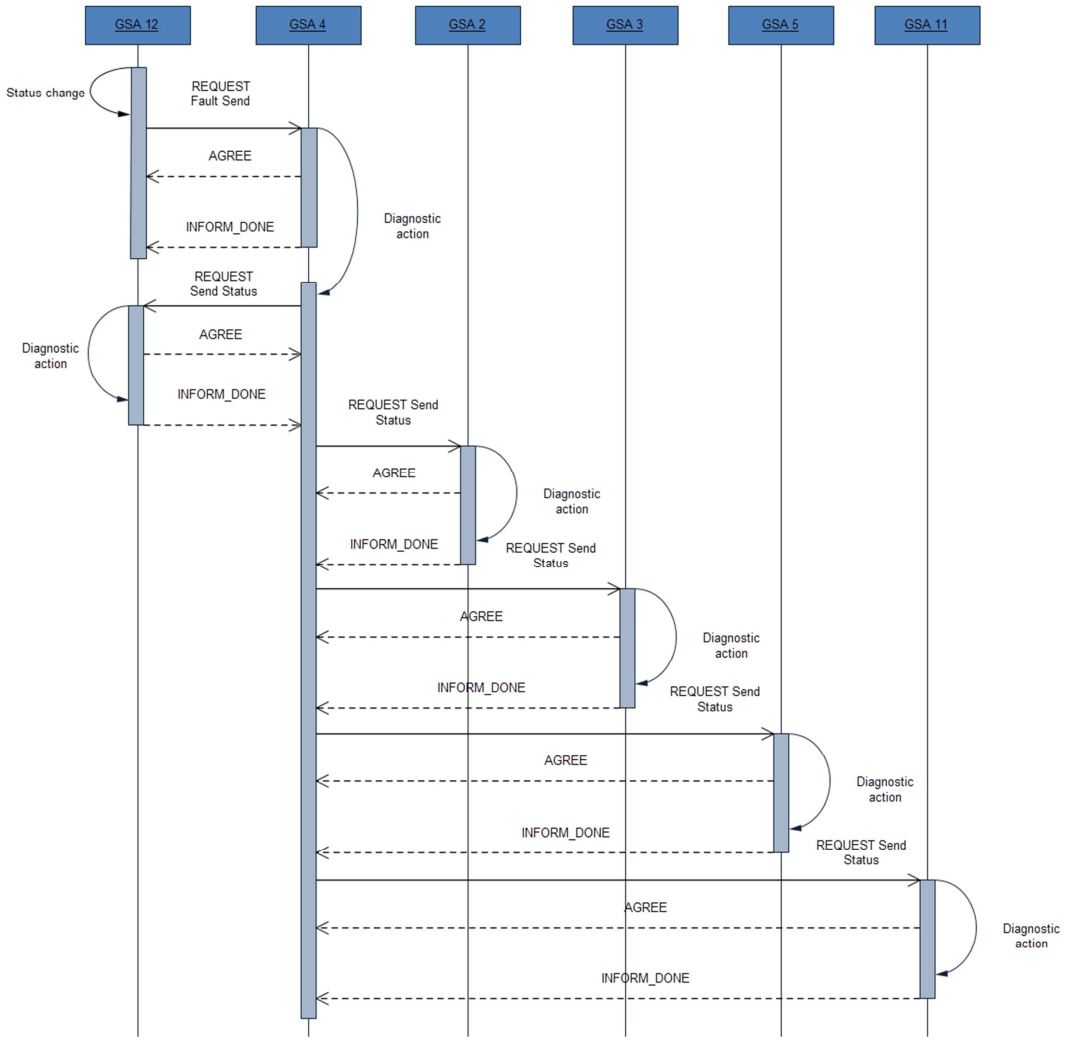


**Figure 31** Message Sequence to inform change in status.

The status propagation behaviour is affected by the matrices modulation, since it is their sensibility to the agent's surroundings together with sensors information that influences the change in the agent status. The objective is to infer the internal agent status through the sensor and surroundings information. If the matrices are too sensitive to the neighbourhood status, a chain reaction might occur in the network. Another factor that affects the status propagation behaviour along with matrices modulation is the network connectivity. A high connectivity network together with matrices sensitivity will potentiate the described effect. The HMM implementation will be detailed in HMM implementation.

Similarly to the Fault Propagation execution, all the diagnostic actions originate the sending of a message to the GDDA, containing the agent name, the fault type and the diagnostic result.

### 4.1.4.1 HMM Implementation

The HMM implementation was based on the jahmm 0.6.1 library [116].

The Hmm implementation started with the definition of diagnostic states. As it was presented in chapter 3, the agent internal states are represented through five states. The Ok state represents the normal and correct function of the device. With this approach two types of faults can be defined, isolated and propagated faults. The NOk state represents an isolated fault, whereas the three remaining states concern to fault propagation. With this classification it is supposed to distinguish and identify both isolated and propagated faults.

It is assumed that Ok is always the initial state which means that λ = [ 1 0 0 0 0 ].



The parameterization of both A and B matrices was achieved using the system tacit knowledge of the author and represents the initial parameterization of each agent diagnostic system. Through the learning algorithm each agent will then adapt itself to its environment, in other words, the agent will parameterize all the matrices according to its observations and will learn to trust more in its sensor or in its neighbourhood.

$$A = \begin{bmatrix} 0.400 & 0.200 & 0.050 & 0.300 & 0.050 \\ 0.050 & 0.500 & 0.400 & 0.000 & 0.050 \\ 0.010 & 0.000 & 0.800 & 0.000 & 0.090 \\ 0.010 & 0.150 & 0.100 & 0.400 & 0.340 \\ 0.005 & 0.010 & 0.180 & 0.005 & 0.800 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.45 & 0.05 & 0.10 & 0.05 & 0.05 & 0.05 & 0.10 & 0.05 & 0.10 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.90 & 0.02 & 0.02 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.43 & 0.33 & 0.00 & 0.05 & 0.05 & 0.00 & 0.07 & 0.07 \\ 0.00 & 0.00 & 0.00 & 0.01 & 0.00 & 0.01 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.56 & 0.00 & 0.00 & 0.42 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.05 & 0.00 & 0.00 & 0.02 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.37 & 0.15 & 0.00 & 0.23 & 0.18 \end{bmatrix}$$

The B matrix was designed to optimize the performance of the diagnostic system. To accomplish the desired behaviour some compromises had to be done. Since both states NOK and PFO are designed to be faults origins, the system was ideally projected to enter in the PFO state only if the agent is the fault propagated origin and in the NOK state only if it is the origin of an isolated fault instead. In the propagation of a fault, the origin agent can feel some repercussions of

the propagated fault, since it can be a neighbour of an affected agent. To avoid the transition from the PFO state to another state, the PFO state was shielded through the A matrix. Figure 32 represents the state transition probability (matrix A). As it is possible to observe the transition probabilities from the PFO are very small, causing the referred shield effect.

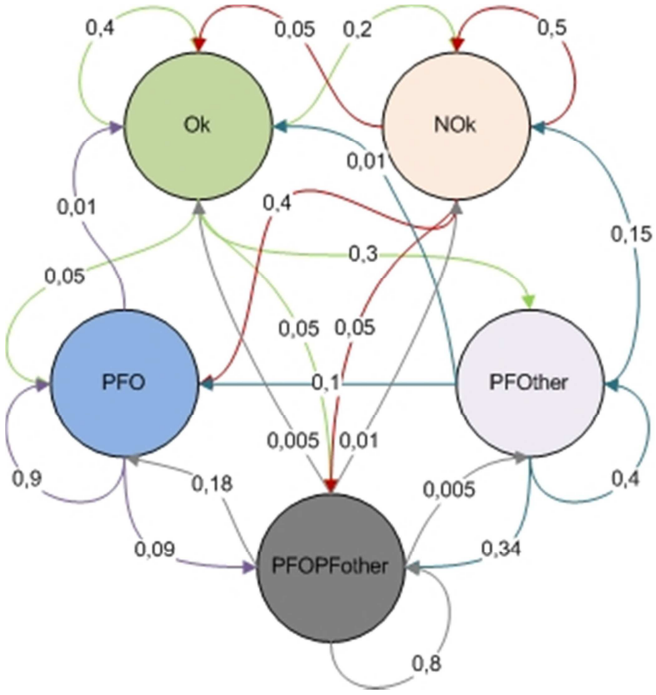


**Figure 32** A matrix representation.

The neighbour's information is only relevant when fault propagation takes place. The initial B matrix was designed to give preference to sensor information. The preference was achieved giving the sensor faulty observations a higher probability, rather than to the ones that didn't present a sensor fault.

As it was stressed before that the B matrix was optimized to the test cases. In order to achieve the best results, the probability of the observations given the state are highly concentrated in only a small number of observations, which gives the system a more sharpen behaviour. It is possible to visualize that parameterization, since in the B matrix, for the propagation states (the last three rows), the probabilities are concentrated in the last observations which are the observations that rely more in the sensor information.

The diagnostic action is performed using the jahmm class ViterbiCalculator. This class implement one of the three characteristics of the HMM. Given the observation O = O1, O2, ..., On, and the model $\lambda = (A, B, \pi)$ devise the state transition sequence that best explains the observations.

### 4.1.4.2 Learning Implementation

As mentioned above the adjustment of the HMM parameters was achieved through resolution of the HMM to the "Problem 3". The adjustment of the model parameters A, B, $\pi$ to maximize the probability of the observation given the model [115], .

The learning mechanism was implemented using the Baum-Welch algorithm. Baum-Welch is a hill-climbing algorithm, which implies that although it finds a better solution on each iteration, it can get stuck on a local maximum. This characteristic makes the results highly dependent on the initial parameterization.



Jahmm library offers two learning classes, BaumWelchScaledLearner and BaumWelchLearner. Both have the same purpose. However, the first overfits the model parameters, while BaumWelchScaledLearner allows the definition of the number of iterations performed by the method.

The learning algorithm was implemented using the class BaumWelchScaledLearner, though an auxiliary function was created in order to calculate the number of iterations until B matrix stabilization. The number of iterations will then be used as reference to perform the scaled learning execution.

Every time a pre-defined number of new faults is observed, the learning mechanism is activated. The user/expert will then be asked to choose one of four learning possibilities regarding the number of iteration. According to the validity of the fault set, the user/expert might want to choose or not a more intensive learning, selecting one quarter, two quarters, three quarters or all of the stabilization iterations value. After the selection, the scaled learning algorithm is executed with the selected iterations number and the model parameters are updated. On the other hand if the fault observations are not relevant the user/expert can cancel the learning execution.

### 4.1.5 Container Manager

The container manager is the interface (Figure 33) that allows the creation and management of containers and respective GSAs. In an agent network the distribution of agents over a number of machines is desired. In order to accomplish component distribution, containers need to be hosted in different machines. Since there can only be a main container in the network all the other containers must join and register in the main container. To simplify the network generation, the container manager allows the creation of both container types.

Regarding the GSAs creation, two methods were implemented, the manual and the automatic GSA generator. The manual GSA generator involves the insertion of the agent name, as well as the manual selection of the xml file, according to the agent pretended functionalities. Furthermore, the automatic GSA generator allows the automatic creation of the defined number of GSAs. The type of GSAs to create can be selected by the user/expert through the radio buttons. The total number of agents created will be equally distributed for each one of the selected types.



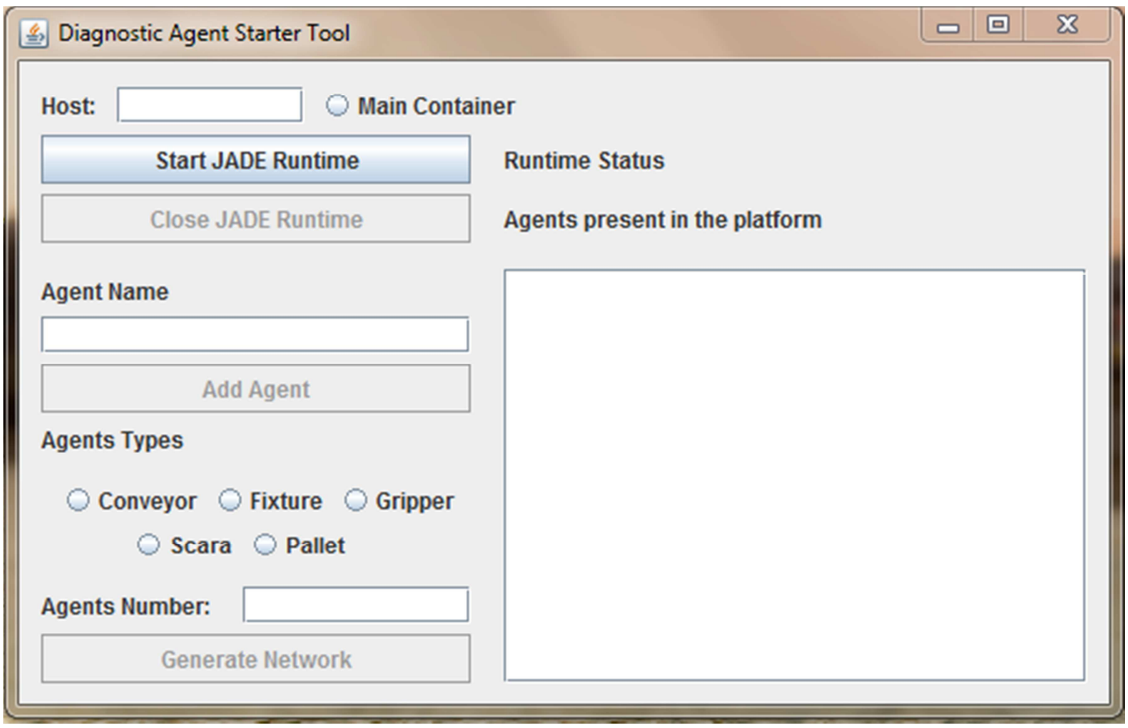


**Figure 33** Container Management interface.

The creation of a new GSA is always reported to the graph diagnostic designer agent (GDDA) through the sequence of messages described in, Figure 34, otherwise the agent will not be represented. The GDDA is simply a graph editor agent that will be further detailed.

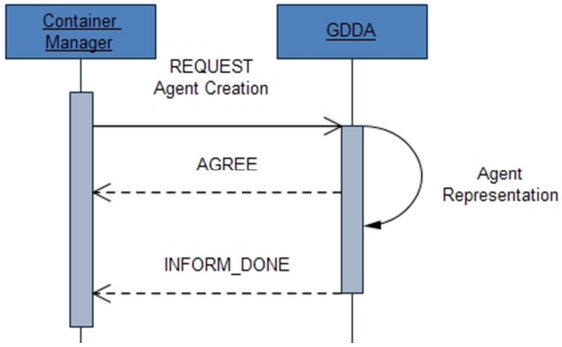


**Figure 34** Message exchange between the Container Manager and the graph diagnostic designer agent.

### 4.1.6 Network Generator Agent Implementation

The Network Generator Agent (NGA), Figure 35, allows the creation of random networks through the insertion of a number of connections or the definition of the network connectivity. Moreover, the NGA also permits the insertion of random faults in the network, the execution of the network diagnostic reset and the definition of the percentage of vulnerable nodes in the network.



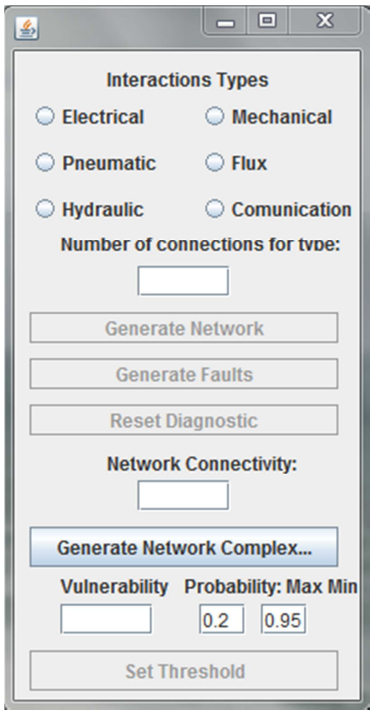


**Figure 35** Network Generator Agent interface.

The role of the NGA in the creation of random networks is limited to the generation of random connections between agents, since the GSAs are created through the Container Manager. The random connections generation consist on selecting randomly two agents, one for the origin and the other for the arrival point, and establishing a selected type link between them. At the same time, a list containing all the generated connections is created in order to avoid the generation of the same connection twice. Whenever a link is established, a message is sent to the GDDA containing the involved agents as well as the type of relation.

The connection creation can be achieved through two different ways: defining the total number of interactions, or inserting the network connectivity. Through the insertion of the interactions number a cycle is performed, and in each iteraction, the previously described mechanism is executed. The second approach implies the generation of a suited adjacency matrix, which is accomplished through the random distribution of the adjacency relations over the entire matrix, except in the main diagonal. This constraint was imposed, since the loops over the same agent have no real representation in this application. After the matrix generation, all the connections are established so that the final network matches the adjacency mapping.



The process of insertion of a random fault in the network consists only in selecting the target agent and establishing a FIPA REQUEST protocol, Figure 36, ordering the simulation of a sensor failure.

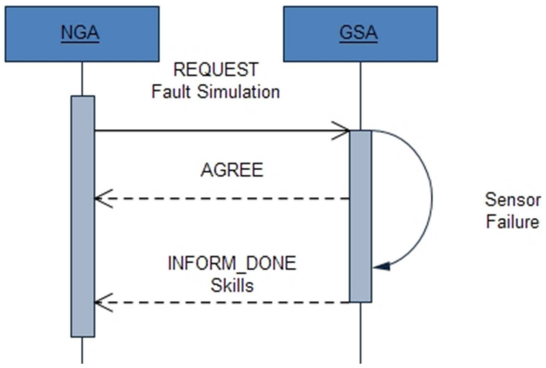


**Figure 36** Message exchange between the Network Generator and the fault simulation target.

Similar to the insertion of a random fault is the execution of the network diagnostic reset, though, instead of only one receiver, the reset message is sent to all agents in the network. When the message is received the actual fault is stored and the diagnostic is re-initiated.

The definition of the network vulnerable agents is achieved by filling the vulnerability text of the NGA interface with the pretended percentage of vulnerable agents in the network and defining the threshold probabilities. Moreover, and according to the inserted probability a number of agents will be randomly selected to be a vulnerable agents. For the vulnerable agents a message will be sent with a high acceptance probability, whereas for the non-vulnerable agents a message will be sent with a low acceptance probability value.

### 4.1.7 Graph Diagnostic Designer Agent

The Graph Diagnostic Designer Agent (GDDA), was implemented through the jung java library (Java Universal Network/Graph Framework), and allows the representation of agent networks graphs. An agent is represented by a node, Figure 37, and the local diagnostic through a colour ring. Each colour represents one internal state of each GSA, Table 8 Diagnostic State representation colours..



| State | Colour |
|---|---|
| Ok | Green |
| NOk | Red |
| PFO | Yellow |
| PFOther | Blue |
| PFOPFOther | Pink |

**Table 8** Diagnostic State representation colours.

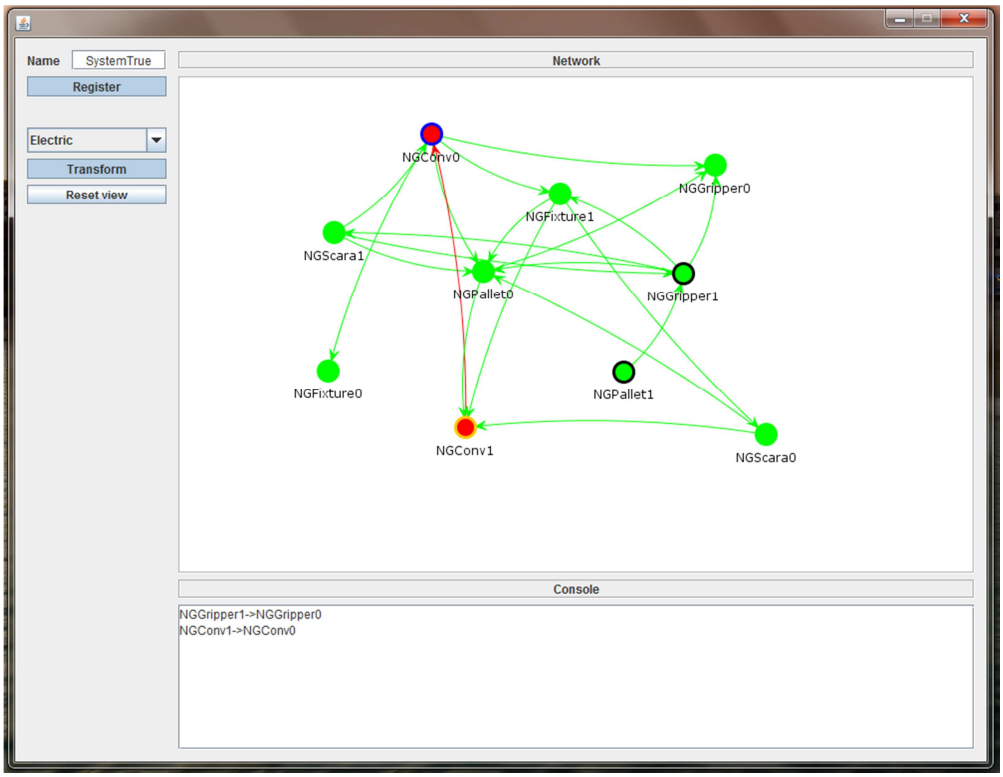




**Figure 37** GDDA interface.

## 4.2 Testing Scenarios and Results Analysis

In order to verify the influence of the number of agents, connectivity and vulnerability in the fault evolution within the network, a set of tests were performed. The tests were conducted under the following specifications:

- Random network topology of 25, 50 and 75 agents (the average number of the agentified NOVAFLEX assembly cell is 25).
- Average degree of connectivity of 1, 2, 3, 6, 9, and 12.
- Vulnerability from 0% of the network to 100% with steps of 5%.
- Fault acceptance for vulnerable and normal agents of (< 20%) and (> 95%) respectively.
- Agent sensor failure percentage of 10%
- 100 faults for vulnerability.

The following charts display the percentage of network affected by faults as vulnerability increases, for the random network topology of 25, 50 and 75 agents respectively.

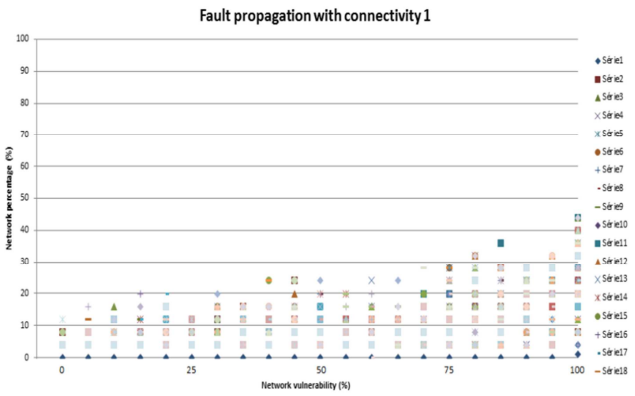


**Chart 1** Fault propagation with average degree of connectivity 1 for a 25 agents random network.

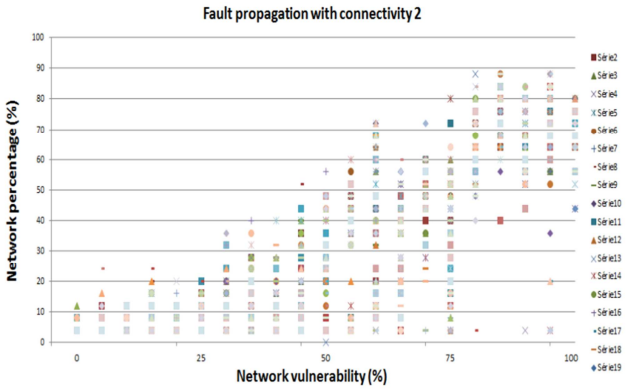


**Chart 2** Fault propagation with average degree of connectivity 2 for a 25 agents random network.

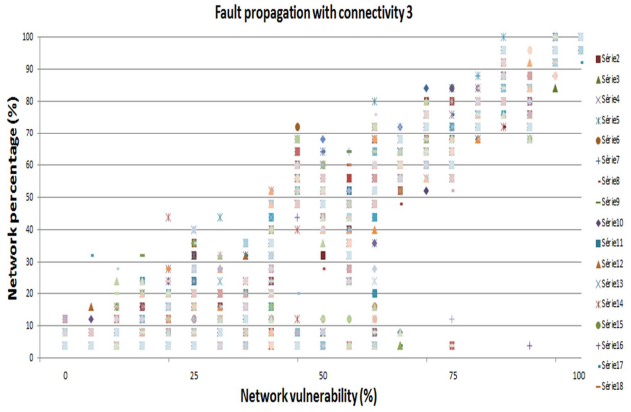


**Chart 3** Fault propagation with average degree of connectivity 3 for a 25 agents random network.

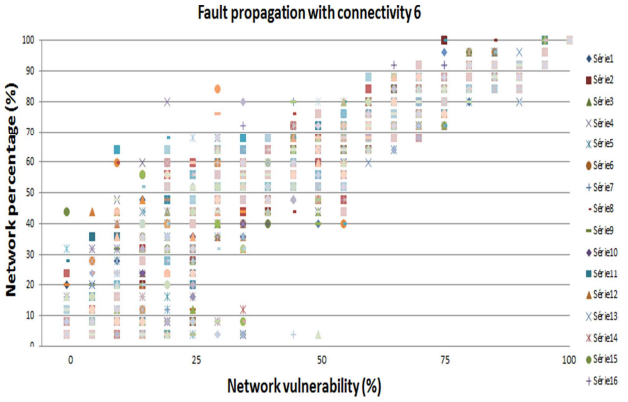


**Chart 4** Fault propagation with average degree of connectivity 6 for a 25 agents random network.

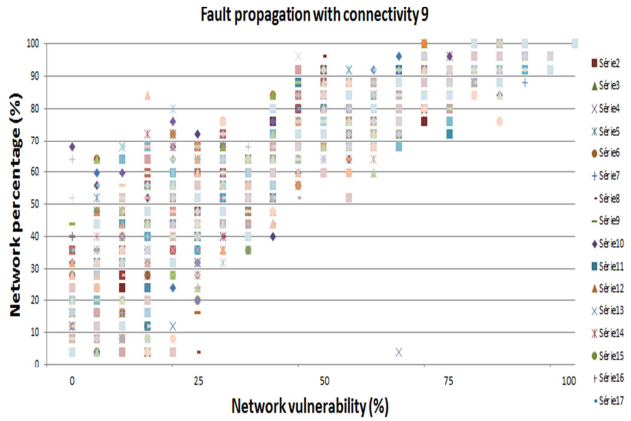


**Chart 5** Fault propagation with average degree of connectivity 9 for a 25 agents random network.

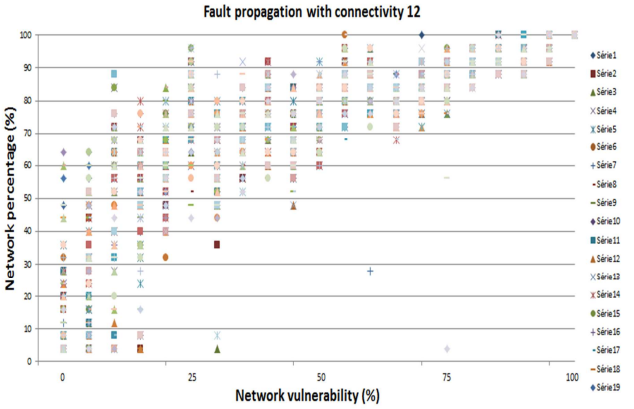


**Chart 6** Fault propagation with average degree of connectivity 12 for a 25 agents random network.

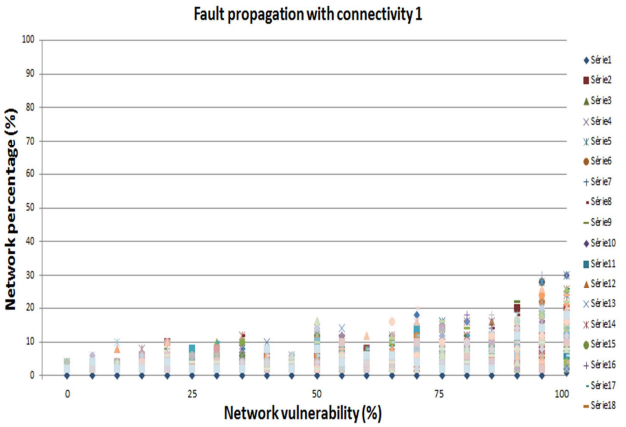


**Chart 7** Fault propagation with average degree of connectivity 1 for a 50 agents random network.

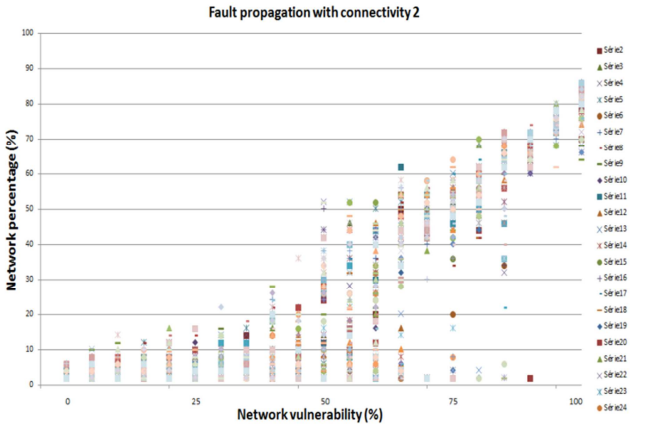


**Chart 8** Fault propagation with average degree of connectivity 2 for a 50 agents random network.

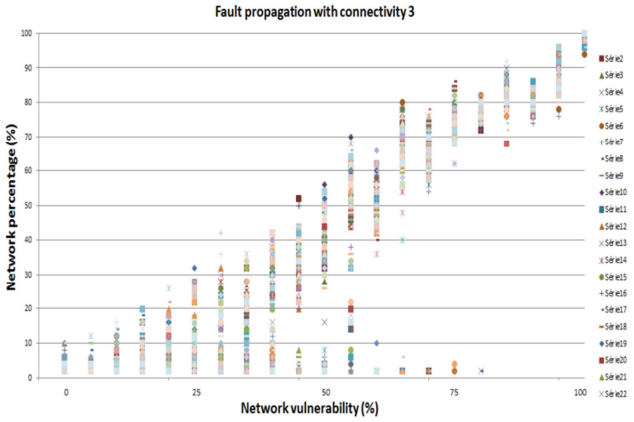


**Chart 9** Fault propagation with average degree of connectivity 3 for a 50 agents random network.

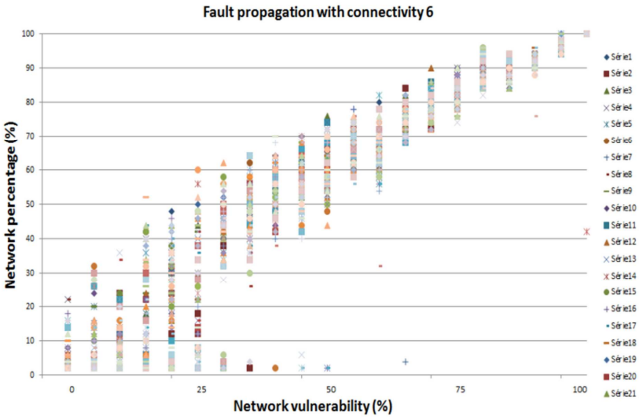


**Chart 10** Fault propagation with average degree of connectivity 6 for a 50 agents random network.

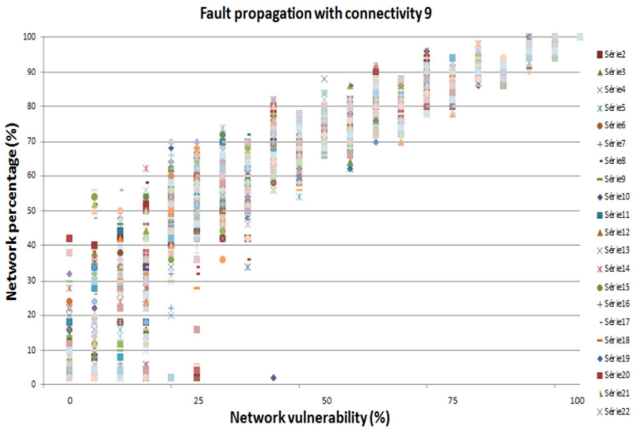


**Chart 11** Fault propagation with average degree of connectivity 9 for a 50 agents random network.

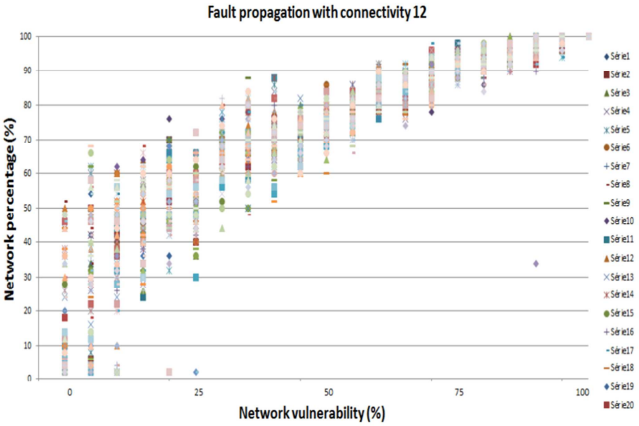


**Chart 12** Fault propagation with average degree of connectivity 12 for a 50 agents random network.

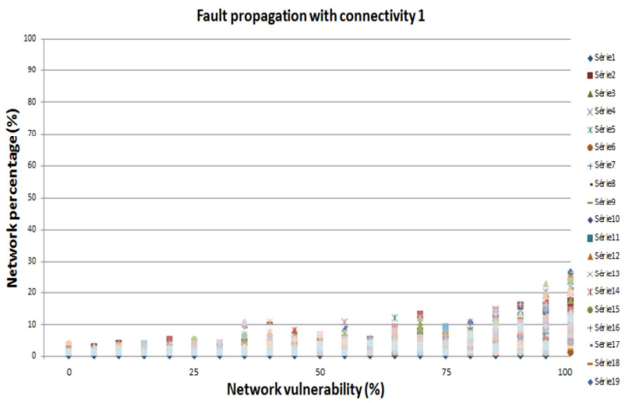


**Chart 13** Fault propagation with average degree of connectivity 1 for a 75 agents random network.

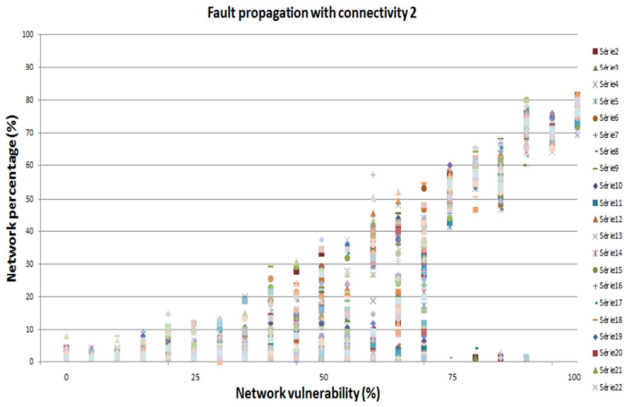


**Chart 14** Fault propagation with average degree of connectivity 2 for a 75 agents random network.

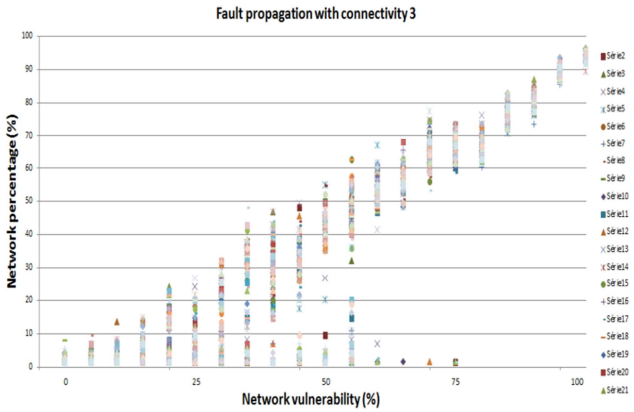


**Chart 15** Fault propagation with average degree of connectivity 3 for a 75 agents random network.

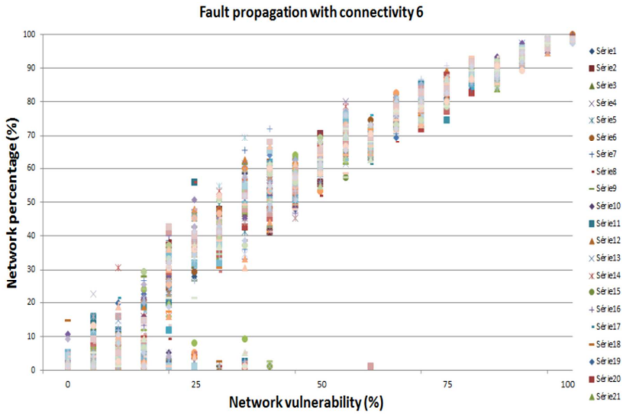


**Chart 16** Fault propagation with average degree of connectivity 6 for a 75 agents random network.



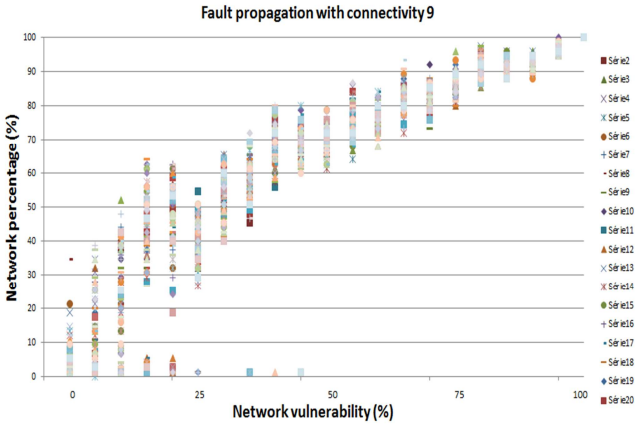


**Chart 17** Fault propagation with average degree of connectivity 9 for a 75 agents random network.

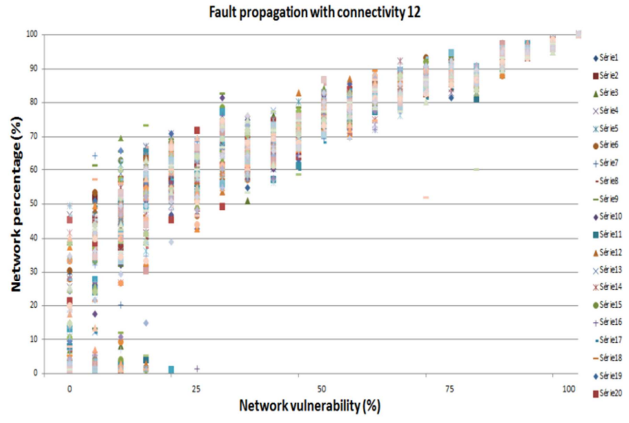


**Chart 18** Fault propagation with average degree of connectivity 12 for a 75 agents random network.

Through the presented sets of charts it is possible to verify that the fault propagation mechanism is independent of the size of the network. This statement is grounded by the similarity presented by the respective connectivity charts from the three sets of tests.

The increase of the fault size with the vulnerability is perceptive in all charts. Nevertheless different evolution patterns are obtained according to the average connectivity degree. It is possible to verify a similar evolution tendency for all sets with the increase of the average connectivity degree. For the low connectivity tests the resulting charts present approximately a linear shape, whereas for the high connectivity the shape is more logarithmic. This behaviour is justified by the impact of vulnerable nodes in fault propagation, which are potentiated by the high connectivity of the networks. Despite the similarities it is possible to observe a narrowing tendency with the increasing size of the networks. Since the propagation system presents similar confidence interval for all sets of tests, the percentage of affected network is lower in bigger networks, which explains this narrowing behaviour.



In order to test the diagnostic system performance in different network conditions, three different groups of tests were performed respecting the previous specifications. The following charts present the results for tests with random networks of 25, 50 and 75 respectively.

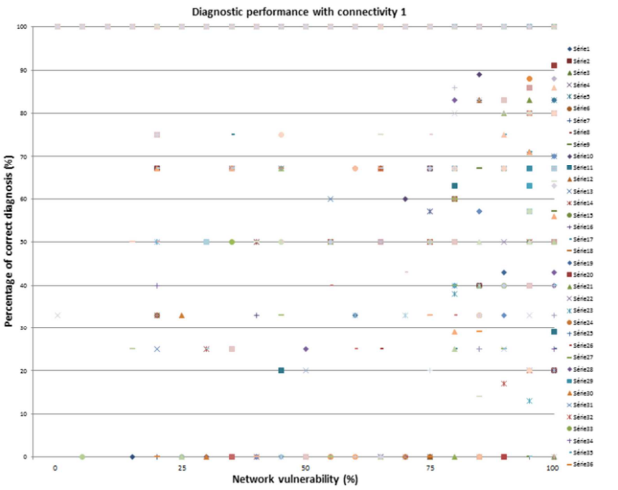


**Chart 19** Diagnostic performance with average connectivity 1 for a 25 agents random network.

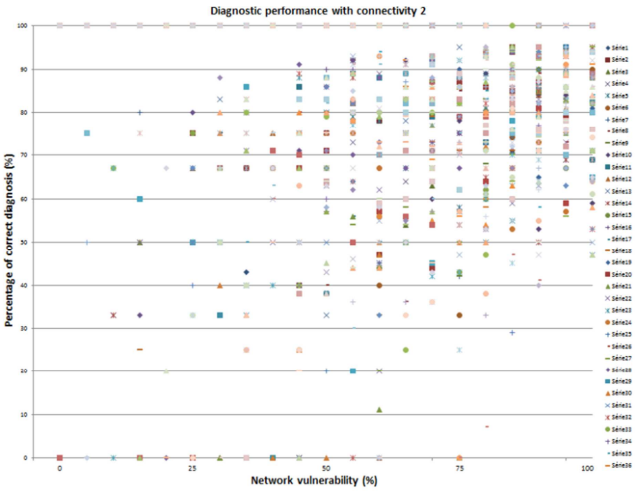


**Chart 20** Diagnostic performance with average connectivity 2 for a 25 agents random network.

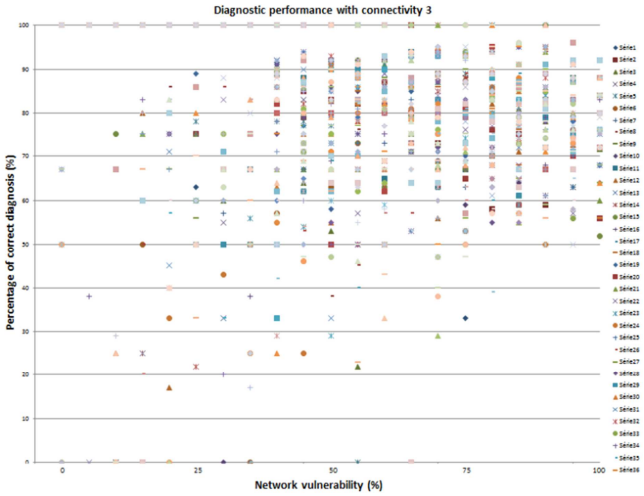


**Chart 21** Diagnostic performance with average connectivity 3 for a 25 agents random network.

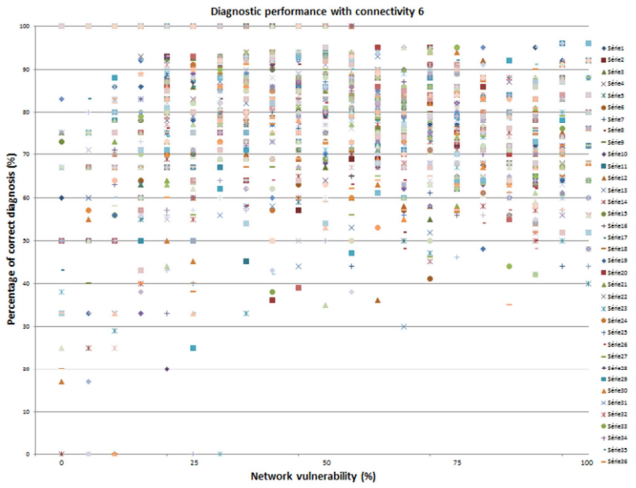


**Chart 22** Diagnostic performance with average connectivity 6 for a 25 agents random network.



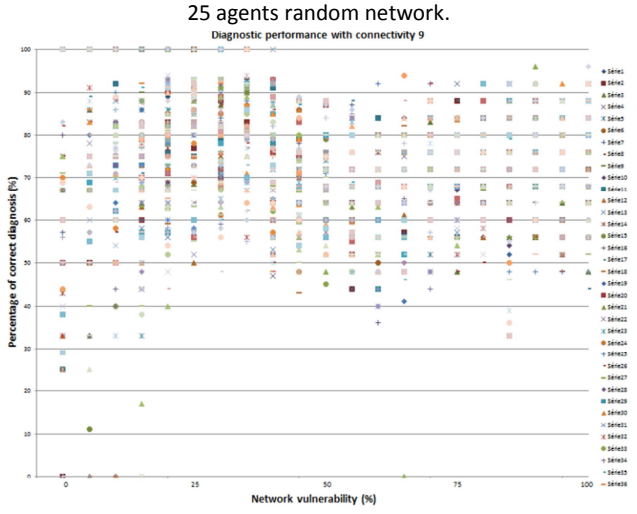


**Chart 23** Diagnostic performance with average connectivity 9 for a 25 agents random network.

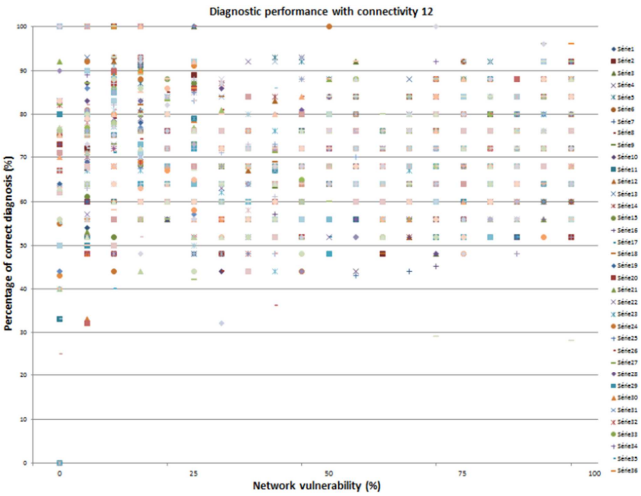


**Chart 24** Diagnostic performance with average connectivity 12 for a 25 agents random network.

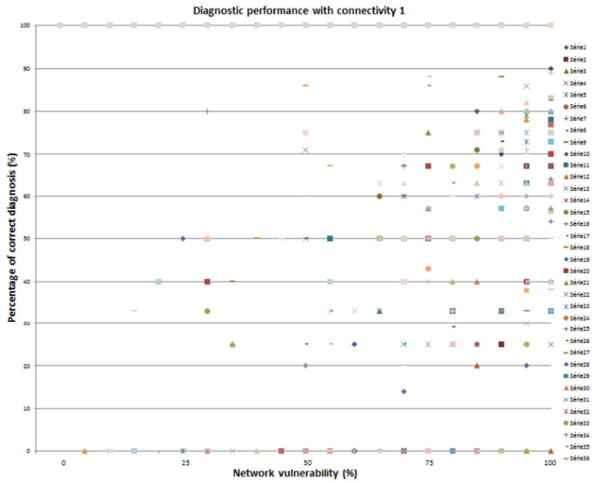


**Chart 25** Diagnostic performance with average connectivity 1 for a 50 agents random network.

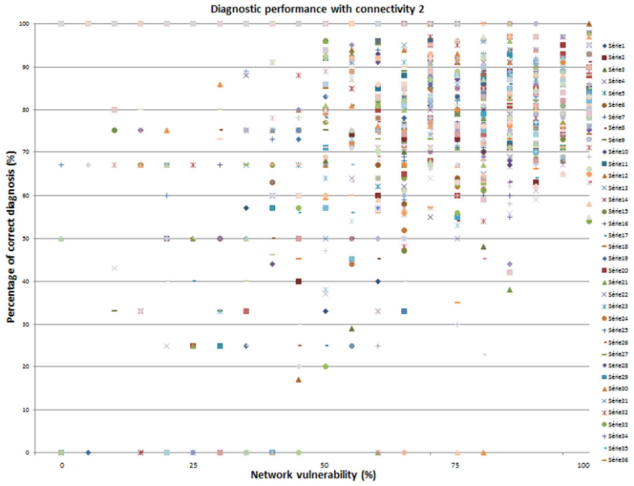


**Chart 26** Diagnostic performance with average connectivity 2 for a 50 agents random network.

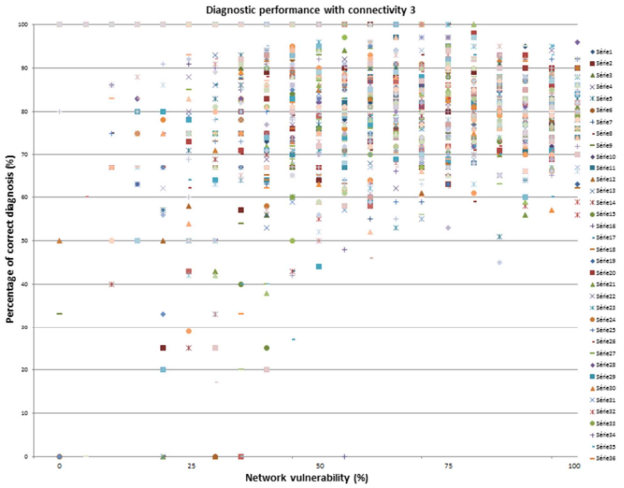


**Chart 27** Diagnostic performance with average connectivity 3 for a 50 agents random network.

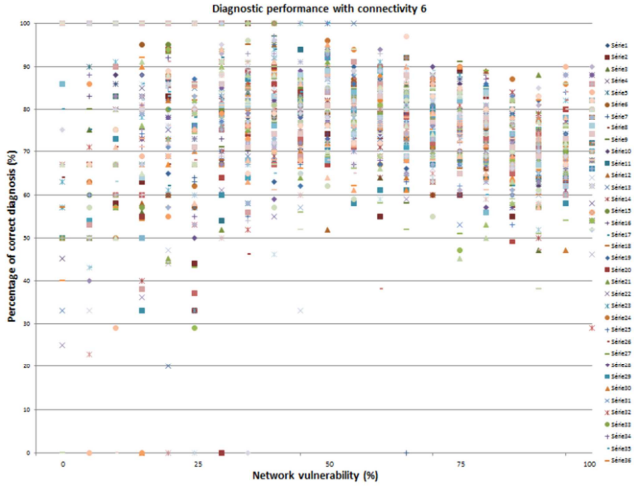


**Chart 28** Diagnostic performance with average connectivity 6 for a 50 agents random network.

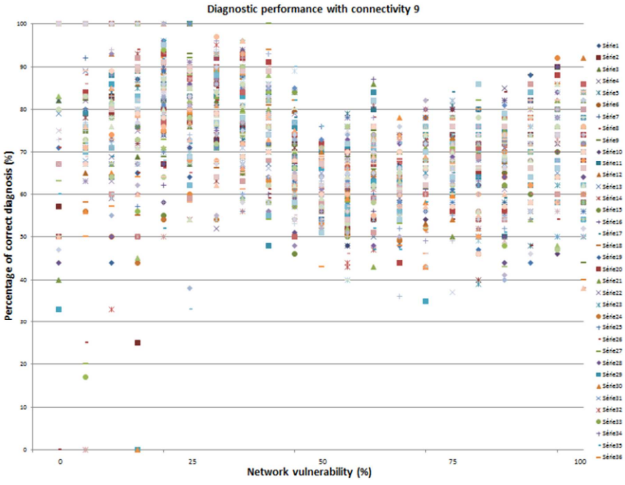


**Chart 29** Diagnostic performance with average connectivity 9 for 50 agents random network.

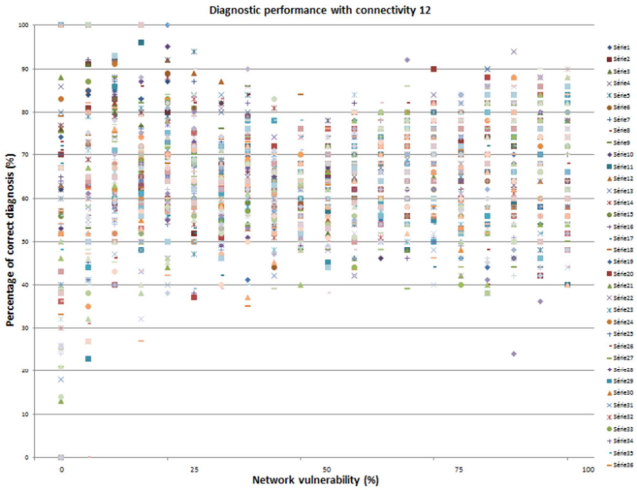


**Chart 30** Diagnostic performance with average connectivity 12 for a 50 agents random network.

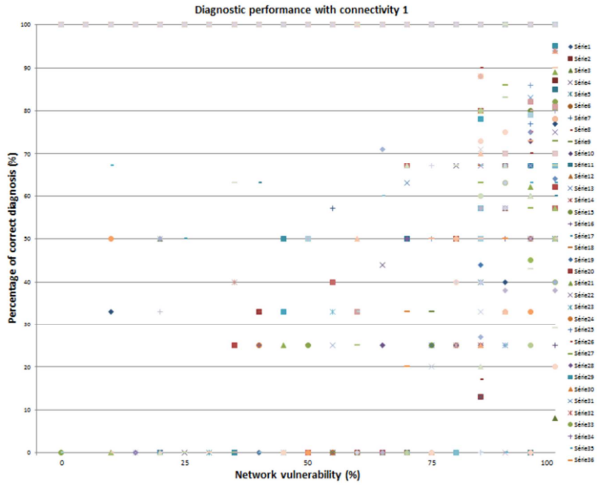


**Chart 31** Diagnostic performance with average connectivity 1 for a 75 agents random network.

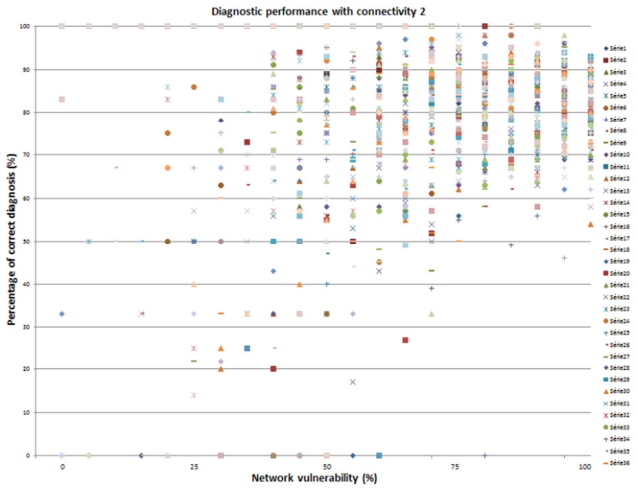


**Chart 32** Diagnostic performance with average connectivity 2 for a 75 agents random network.

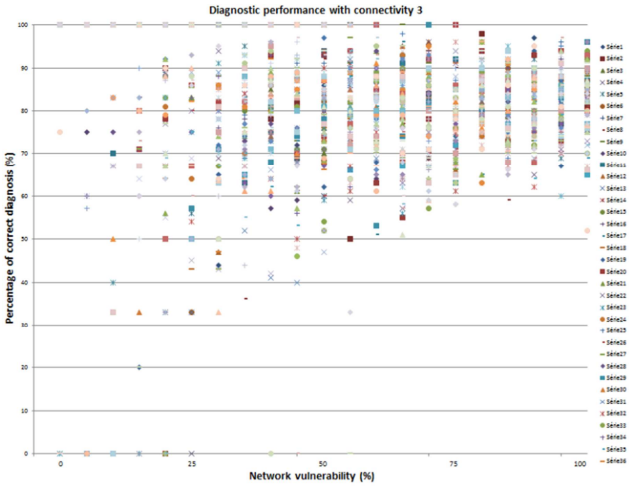


**Chart 33** Diagnostic performance with average connectivity 3 for a 75 agents random network.

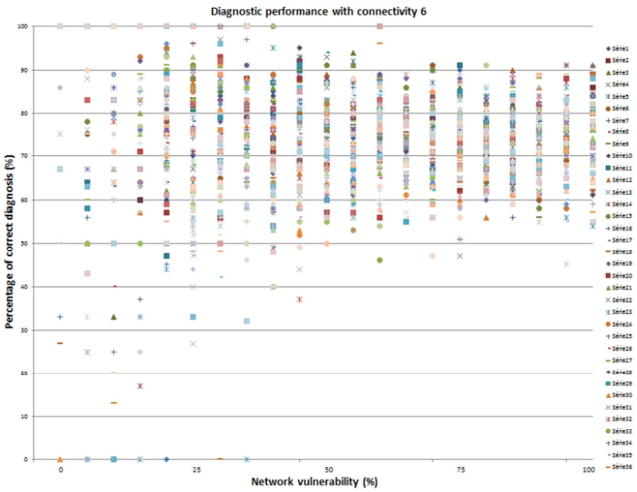


**Chart 34** Diagnostic performance with average connectivity 6 for a 75 agents random network.

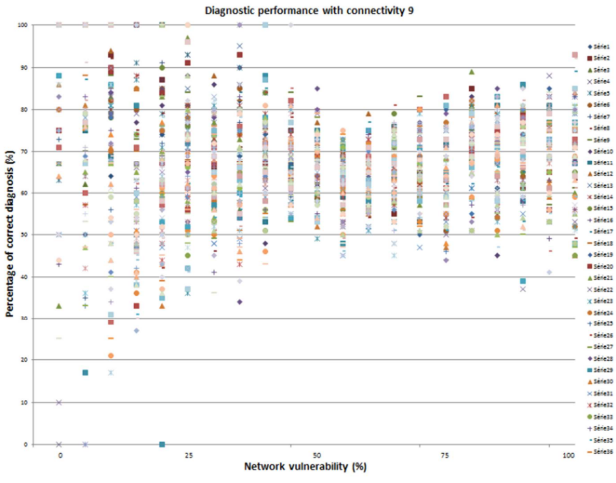


**Chart 35** Diagnostic performance with average connectivity 9 for a 75 agents random network.

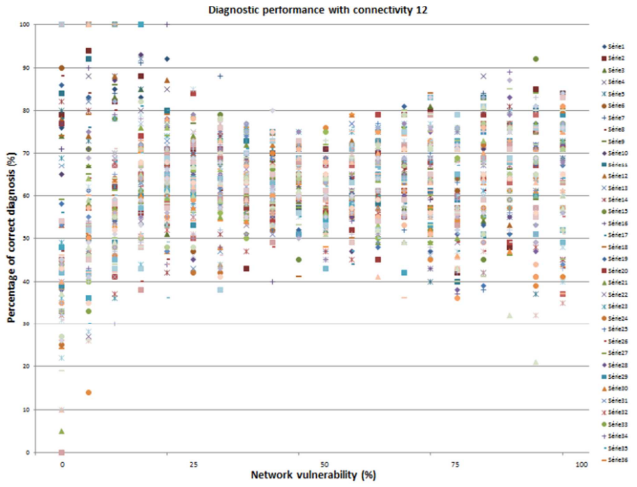


**Chart 36** Diagnostic performance with average connectivity 12 for a 75 agents random network.

Through the analysis of the three different sets of tests and due to their similarity was possible to conclude that the diagnostic system was not affected by the number of agents in the network. This was an expected result, since the propagation mechanism was not size dependent. Therefore, was likely that the diagnostic performance would remain, independently of the size of the network.

Despite the similarities of the diagnostic performance between different network topologies, it is possible to perceive that the diagnostic system is affected by the average connectivity degree as well as by the vulnerability. For the lower connectivity charts the dispersion cloud is less dense, since for the lower vulnerabilities the diagnostic performance is in the majority of the faults 100%. This performance decreases with the vulnerability which implicates the augmentation of the dispersion. The diagnostic system is less efficient in the fault end agents, once they had to identify their terminal position in the propagation. Therefore with higher vulnerabilities and consequently bigger faults propagations the diagnostic system presents a small performance decrease.



For the higher connectivity charts, a particular behaviour can be identified. In Chart 23, 24, 29, 30, 35, 36 it is possible to verify a narrowing tendency of the dispersion cloud which is again expanded for higher vulnerabilities. This unexpected behaviour for medium vulnerability percentages can be justified through the high ramification of some faults in the network and the less efficiency of the diagnostic system leading with this type of faults. Moreover, for the high vulnerabilities the faults effects are extended to the entire network. Therefore the diagnostic performance improves, since the majority of the agents are receiving and propagating the fault.

The relation between diagnostic performance and average connectivity degree is depicted in Charts [37; 42]. From the presented tests, two vulnerabilities percentages of each average connectivity degree were chosen to represent a low and a highly connected network, 10% and 70 % respectively.

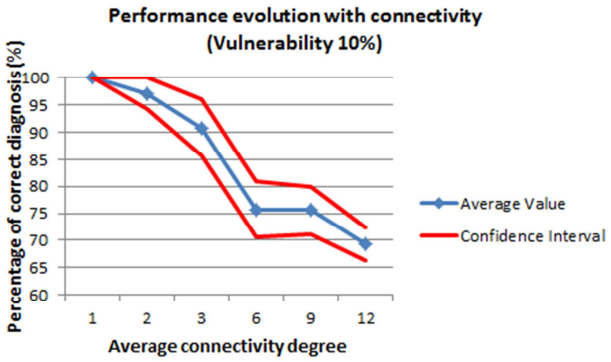


**Chart 37** Diagnostic performance for different connectivity degrees and with 10% of vulnerability for a 25 random network.

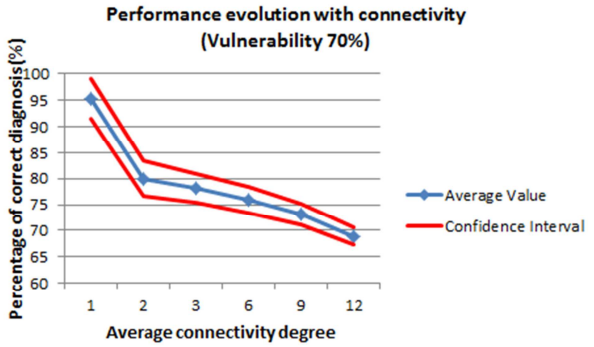


**Chart 38** Diagnostic performance for different connectivity degrees and with 70% of vulnerability for a 25 random network.

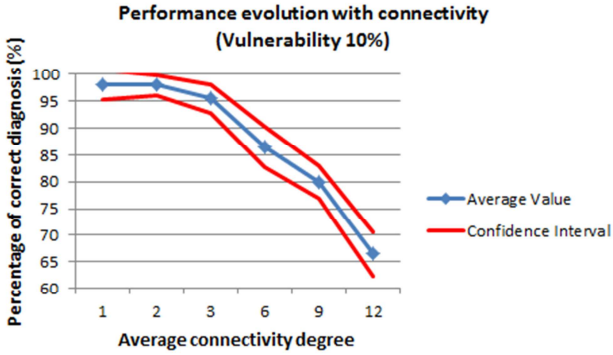


**Chart 39** Diagnostic performance for different connectivity degrees and with 10% of vulnerability for a 50 random network.

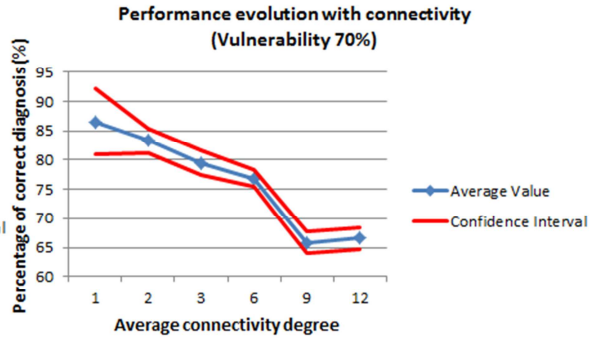


**Chart 40** Diagnostic performance for different connectivity degrees and with 70% of vulnerability for a 50 random network.



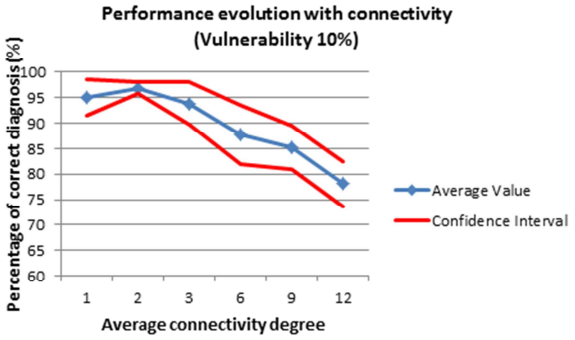


**Chart 41** Diagnostic performance for different connectivity degrees and with 10% of vulnerability for a 75 random network.

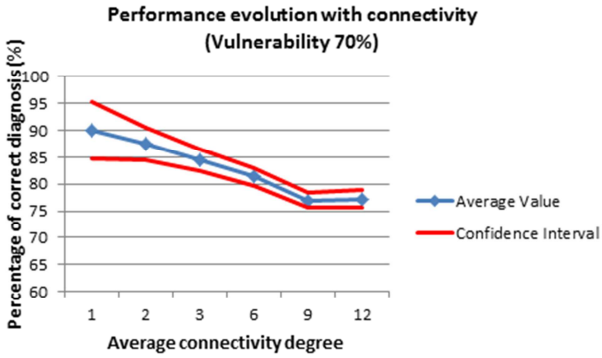


**Chart 42** Diagnostic performance for different connectivity degrees and with 70% of vulnerability for a 75 random network.

For each test the standard deviation was computed and used to calculate the confidence interval with a confidence level of 95 %. The confidence interval is represented in the charts through the red lines, and presents values from 0 to 5.8. Although perceptible in the former tests, the effect of the average connectivity degree in the performance becomes clearer through the analysis of the above charts. The average connectivity degree has a negative impact in the

diagnostic performance. Only one more neighbour can represent an all new set of scenarios for which the diagnostic system may not be parameterized.

More particularly, for the 10% vulnerability charts, it is observable that the graph has approximately a more parabolic characteristic against the 70% vulnerability that present a more linear tendency. This tendency reveals that low vulnerable networks are more connectivity proof because the diagnostic system performance is only affected by high average connectivity degrees networks. In highly connected networks it is possible to observe that for an average connectivity degree of two the diagnostic system performance was significantly deteriorated.

Regarding the statistical analysis, when the data to be analysed are derived by counting the number of positive outcomes of repeated identical and independent experiments, it can be considered a binomial distribution [117]. Therefore each test defines an independent binomial distribution. Considering that each fault is an independent test then a fault can be thought has a Bernoulli trial. A Bernoulli trial is the statistical idealization of a coin flip in which there is a fixed probability of a successful outcome that does not vary from flip to flip [117].Therefore the tests were thought to satisfy the condition (variance = np(1-p)) $np(1-p) \geq 5$ which ensures the approximation of the binomial distribution by the normal distribution.



The performed tests proved the system performance diagnosing random failures in random networks, which can be considered the worst case scenario. To test the diagnostic system performance in real applications, the NOVAFLEX manufacturing cell was used as test scenario.

The test consisted in a set of 100 logical and probable faults. The fault acceptance along with other probabilities and test variables were neutralized, since the faults were manually introduced in the system.

The Table 9 presents the NOVAFLEX tests results.

| Fault nº | Number of diagnoses | Correct diagnoses | Failed diagnoses | Correct diagnoses (%) |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 100 |
| 2 | 2 | 2 | 0 | 100 |
| 3 | 3 | 3 | 0 | 100 |
| 4 | 4 | 4 | 0 | 100 |

| | | | | |
|---|---|---|---|---|
| 5 | 5 | 5 | 0 | 100 |
| 6 | 6 | 6 | 0 | 100 |
| 7 | 7 | 7 | 0 | 100 |
| 8 | 8 | 8 | 0 | 100 |
| 9 | 7 | 7 | 0 | 100 |
| 10 | 8 | 7 | 1 | 88 |
| 11 | 6 | 6 | 0 | 100 |
| 12 | 7 | 7 | 0 | 100 |
| 13 | 8 | 8 | 0 | 100 |
| 14 | 9 | 8 | 1 | 89 |
| 15 | 9 | 8 | 1 | 89 |
| 16 | 8 | 8 | 0 | 100 |
| 17 | 9 | 9 | 0 | 100 |
| 18 | 10 | 9 | 1 | 90 |
| 19 | 10 | 10 | 0 | 100 |
| 20 | 9 | 9 | 0 | 100 |
| 21 | 11 | 10 | 1 | 91 |
| 22 | 12 | 10 | 2 | 83 |
| 23 | 1 | 1 | 0 | 100 |
| 24 | 2 | 2 | 0 | 100 |
| 25 | 3 | 3 | 0 | 100 |
| 26 | 1 | 1 | 0 | 100 |
| 27 | 2 | 2 | 0 | 100 |
| 28 | 3 | 3 | 0 | 100 |
| 29 | 5 | 5 | 0 | 100 |
| 30 | 4 | 4 | 0 | 100 |
| 31 | 4 | 4 | 0 | 100 |
| 32 | 3 | 3 | 0 | 100 |
| 33 | 2 | 2 | 0 | 100 |
| 34 | 2 | 2 | 0 | 100 |
| 35 | 1 | 1 | 0 | 100 |
| 36 | 4 | 4 | 0 | 100 |
| 37 | 3 | 3 | 0 | 100 |
| 38 | 3 | 3 | 0 | 100 |
| 39 | 2 | 2 | 0 | 100 |
| 40 | 6 | 6 | 0 | 100 |
| 41 | 7 | 6 | 1 | 86 |
| 42 | 3 | 2 | 1 | 67 |
| 43 | 3 | 3 | 0 | 100 |
| 44 | 4 | 4 | 0 | 100 |
| 45 | 4 | 1 | 3 | 25 |
| 46 | 4 | 4 | 0 | 100 |
| 47 | 6 | 6 | 0 | 100 |
| 48 | 4 | 0 | 4 | 0 |
| 49 | 3 | 3 | 0 | 100 |
| 50 | 5 | 4 | 1 | 80 |
| 51 | 1 | 1 | 0 | 100 |
| 52 | 2 | 2 | 0 | 100 |
| 53 | 3 | 3 | 0 | 100 |
| 54 | 4 | 4 | 0 | 100 |
| 55 | 5 | 5 | 0 | 100 |
| 56 | 6 | 6 | 0 | 100 |

| | | | | |
|---|---|---|---|---|
| 57 | 7 | 7 | 0 | 100 |
| 58 | 8 | 8 | 0 | 100 |
| 59 | 7 | 7 | 0 | 100 |
| 60 | 8 | 7 | 1 | 88 |
| 61 | 6 | 6 | 0 | 100 |
| 62 | 7 | 7 | 0 | 100 |
| 63 | 8 | 8 | 0 | 100 |
| 64 | 9 | 8 | 1 | 89 |
| 65 | 9 | 8 | 1 | 89 |
| 66 | 8 | 8 | 0 | 100 |
| 67 | 9 | 9 | 0 | 100 |
| 68 | 10 | 9 | 1 | 90 |
| 69 | 10 | 10 | 0 | 100 |
| 70 | 9 | 9 | 0 | 100 |
| 71 | 11 | 10 | 1 | 91 |
| 72 | 12 | 10 | 2 | 83 |
| 73 | 1 | 1 | 0 | 100 |
| 74 | 2 | 2 | 0 | 100 |
| 75 | 3 | 3 | 0 | 100 |
| 76 | 1 | 1 | 0 | 100 |
| 77 | 2 | 2 | 0 | 100 |
| 78 | 3 | 3 | 0 | 100 |
| 79 | 5 | 5 | 0 | 100 |
| 80 | 4 | 4 | 0 | 100 |
| 81 | 4 | 4 | 0 | 100 |
| 82 | 3 | 3 | 0 | 100 |
| 83 | 2 | 2 | 0 | 100 |
| 84 | 2 | 2 | 0 | 100 |
| 85 | 1 | 1 | 0 | 100 |
| 86 | 4 | 4 | 0 | 100 |
| 87 | 3 | 3 | 0 | 100 |
| 88 | 3 | 3 | 0 | 100 |
| 89 | 2 | 2 | 0 | 100 |
| 90 | 6 | 6 | 0 | 100 |
| 91 | 7 | 6 | 1 | 86 |
| 92 | 3 | 2 | 1 | 67 |
| 93 | 3 | 3 | 0 | 100 |
| 94 | 4 | 4 | 0 | 100 |
| 95 | 4 | 1 | 3 | 25 |
| 96 | 4 | 4 | 0 | 100 |
| 97 | 6 | 6 | 0 | 100 |
| 98 | 4 | 0 | 4 | 0 |
| 99 | 3 | 3 | 0 | 100 |
| 100 | 5 | 4 | 1 | 80 |
| Total | 303 | 281 | 22 | 93.17 |

**Table 9** Tests results from the diagnostic system in the NOVAFLEX manufacturing cell.

As it is possible to observe in the majority of the faults the diagnostic performance is about 100%, presenting just one case with 0% of correctness. In order to better understand the achieved results some statistical values will be presented in Table 10.

| | |
|---|---|
| Average | 93.76 |
| Count | 100 |
| Standard deviation | 18.1 |
| Confidence interval | 3.55 |
| np(1-p) | 5.85 |

**Table 10** Statistical results from NOVAFLEX data tests.

As it was expected the results present a good performance of around 93% of correct diagnostics. The Standard deviation of the set of test was of 18.19, which led to a 95 % confidence interval of 3.55. The variance value of the test is 5.85 which allow the approximation by the normal distribution.



The learning mechanism was fully implemented as it was described in the former chapters. However, due to test difficulties was not possible to perform sufficient tests to prove its efficiency. Consequently the learning mechanism was not validated.

# 5 CONCLUSIONS AND FUTURE WORK

*In this chapter conclusions from the developed work as well as some proposals for future work are presented.*

## 5.1 Conclusions

New production paradigms and trends are increasingly relying in distributed approaches to address new challenges and to cope with new market and economical demands. As systems become increasingly decoupled and intelligent, distributed and complex traditional diagnostic will become insufficient.



Diagnostic is usually pointed out in literature as a crucial base to economic and environmental sustainability. Nonetheless diagnostic is essential as a regulatory mechanism in complex distributed systems. Current diagnostic research and literature is incredibly vast, however, it is very focused to the device level. The research efforts concerning the impact of interactions between different intelligent components and modules which can potentiate the propagation of faults that cannot be interpreted only at the device level, has been little or none.

The agent technology has proved to be a powerful tool in the development of a distributed control and diagnostic system meeting all the EPS paradigm requisites. The importance of this technology in the state-of-the-art computational system is increasing following the change of computing landscape from a focus on individual computer system, to a reality were the real computing power will be achieved through distribution.

The implementation of the MAS EPS compliant generic control architecture represented a step forward in the research of generic architectures based on MAS. In this context, must be highlighted the generic nature of the agents and the capacity of abstraction multiple components without any reprogramming. The orchestrator nature of the each intelligent individual also potentiates the fault tolerance of the system, since local faults can be overcome without any impact to the rest of the system. The capacity to deal with faults at a local and system level is also important to stress. One of the major downsize of this implementation is its footprint which implies the necessity of some computation power at device level.

Furthermore the development of a diagnostic system which provides a fully distributed and seamlessly pluggable solution for sustainable production environments was successfully achieved as the presented results confirm In total were simulated around 38000 faults, which can assure some relevance in the obtained results. The HMM proved to be adequate to distributed system and more particularly to EPS compliant architectures. Despite the non-validation of the learning mechanism its simplicity is also a positive and important characteristic. On the other hand, the dependence on initial conditions implies a historic knowledge of the system by the user.



Some preliminary results as well as the architecture supporting this work were submitted and accepted, in paper format, in the following peer-reviewed international conferences:

- 1º Doctoral Conference on Computing, Electrical and Industrial Systems [113].
- International Symposium on Industrial Electronics [118].
- International Conference on Industrial Informatics [107] (Best Paper Award).
- Workshop on Intelligent Manufacturing Systems [96].
- International Conference on Self-Adaptive and Self-Organizing Systems [119].
- 2º Doctoral Conference on Computing, Electrical and Industrial Systems (waiting for approval).

Other publications are being prepared to expose the diagnostic system final results.

It's the author belief that the developed work presents a new and innovative approach in such a vast domain as the diagnostic research field. Through the implementation of this work was also possible to reinforce the suitability of MAS architecture to distributed computing systems.

## 5.2 Future Work

One of the main challenges to be tackled as future work is the validation of the learning mechanism. The development of similar architectures for mobile and low power devices would also be interesting. Therefore efforts should be made in order to reduce the generic agent footprint. A more user friendly workflow generator should also be developed due to the implicit difficulty of manually defining an xml file.

Finally but also interesting would be the development of an algorithm able to properly create the HMM model through the online capture of the system behaviour. Similarly, efforts should be made to improve the learning system in order to provide an automatic adaptation of the HMM to system changes, without the consulting of the user/expert.

# 6 REFERENCES


**[1]** **V. Venkatasubramanian, *et al.*, "A review of process fault detection and diagnosis:: Part I: Quantitative model-based methods," *Computers & Chemical Engineering*, vol. 27, pp. 293-311, 2003.**

**[2]** **P. Young, *et al.*, "Manufacturing and the Environment," *The International Journal of Advanced Manufacturing Technology*, vol. 13, pp. 488-493, 1997.**

**[3]** **H. Wörn, *et al.*, "DIAMOND: distributed multi-agent architecture for monitoring and diagnosis," *Production Planning & Control*, vol. 15, pp. 189-200, 2004.**

**[4]** **G. Deconinck, *et al.*, "Distributed embedded automation systems: dynamic environments and dependability," 2001.**

**[5]** **R. C. Scott de Deugd, Kevin Kelly, Bill Millett and Jeffrey Ricker, "SODA: Service Oriented Device Architecture," 2006.**

**[6]** **L. de Souza, *et al.*, "Socrades: A web service based shop floor integration infrastructure," *The Internet of Things*, pp. 50-67, 2008.**

**[7]** **H. Bohn, *et al.*, "SIRENA-Service Infrastructure for Real-time Embedded Networked Devices: A service oriented framework for different domains," 2006.**

**[8]** ***EUPASS : Evolvable Ultra-Precision Assembly Systems*. Available: http://cordis.europa.eu/fetch?CALLER=PROJ_ICT&ACTION=D&CAT=PROJ&RCN=75342**

**[9]** **A. Giua, "Distributed Supervisory Control of Large Plants," ed. Brussels, 2008.**

**[10]** **F. Bellifemine, *et al.*, "Java Agent Development Framework," *TILAB Italia, http://jade. cselt. it, status*, vol. 10, p. 2002, 2002.**

**[11]** **F. Bellifemine, *et al.*, "JADE—a java agent development framework," *Multi-Agent Programming*, pp. 125-147, 2005.**

**[12]** **A. Sethi and S. Sethi, "Flexibility in manufacturing: a survey," *International Journal of Flexible Manufacturing Systems*, vol. 2, pp. 289-328, 1990.**

**[13]** **R. Johnstone and J. Kurtzhaltz, "Flexible manufacturing system," ed: Google Patents, 1984.**

**[14]** **L. Sanchez and R. Nagi, "A review of agile manufacturing systems," *International Journal of Production Research*, vol. 39, pp. 3561-3600, 2001.**

**[15]** **J. Barata, "Coalition based approach for shopfloor agility," 2005.**

**[16]** **B. Maskell, "The age of agile manufacturing," *Supply Chain Management: An International Journal*, vol. 6, pp. 5-11, 2001.**

**[17]** **R. Babiceanu and F. Chen, "Development and applications of holonic manufacturing systems: a survey," *Journal of Intelligent Manufacturing*, vol. 17, pp. 111-131, 2006.**

**[18]** **P. Valckenaers, *et al.*, "Holonic manufacturing systems," *Integrated Computer-Aided Engineering*, vol. 4, pp. 191-201, 1997.**

**[19]** **S. Bussmann and D. Mcfarlane, "Rationales for holonic manufacturing control," 1999, pp. 177-184.**

**[20]** **P. Leitão, "Agent-based distributed manufacturing control: A state-of-the-art survey," *Engineering Applications of Artificial Intelligence*, vol. 22, pp. 979-991, 2009.**

**[21]** **L. Camarinha-Matos, "New collaborative organizations and their research needs," *Processes and Foundations for Virtual Organizations*, pp. 3-12, 2004.**

**[22]** **L. Camarinha-Matos and H. Afsarmanesh, "Collaborative networks: A new scientific discipline," *Journal of Intelligent Manufacturing*, vol. 16, pp. 439-452, 2005.**

**[23]** **Y. Koren, *et al.*, "Reconfigurable manufacturing systems," *CIRP Annals-Manufacturing Technology*, vol. 48, pp. 527-540, 1999.**

**[24]** **M. Mehrabi, *et al.*, "Reconfigurable manufacturing systems: key to future manufacturing," *Journal of Intelligent Manufacturing*, vol. 11, pp. 403-419, 2000.**

**[25]** **M. Mehrabi, *et al.*, "Trends and perspectives in flexible and reconfigurable manufacturing systems," *Journal of Intelligent manufacturing*, vol. 13, pp. 135-146, 2002.**

**[26]** **J. Bollinger, *et al.*, "Visionary manufacturing challenges for 2020," *National Research Council Report*, 1998.**

**[27]** **P. Heytler and A. Ulsoy, "A Survey of Flexible and Reconfigurable Manufacturing Systems (RMSs)," Internal Report, Engineering Research Center for Reconfigurable Machining Systems (ERC/RMS), The University of Michigan, Ann Arbor1997.**

**[28]** **F. Maturana, *et al.*, "MetaMorph: an adaptive agent-based architecture for intelligent manufacturing," *International Journal of Production Research*, vol. 37, pp. 2159-2173, 1999.**

**[29]** **S. Kumar and P. Cohen, "Towards a fault-tolerant multi-agent system architecture," 2000, pp. 459-466.**

**[30]** **R. Flores-Mendez, "Towards a standardization of multi-agent system framework," *Crossroads*, vol. 5, pp. 18-24, 1999.**

**[31]** **P. Leitão and F. Restivo, "ADACOR: a holonic architecture for agile and adaptive manufacturing control," *Computers in Industry*, vol. 57, pp. 121-130, 2006.**

**[32]** **L. Monostori, *et al.*, "Agent-based systems for manufacturing," *CIRP Annals-Manufacturing Technology*, vol. 55, pp. 697-720, 2006.**

**[33]** **M. Onori, *et al.*, "Evolvable assembly systems basic principles," *Information Technology for Balanced Manufacturing Systems*, pp. 317-328, 2006.**

**[34]** **J. Barata, *et al.*, "Evolvable Production Systems: Enabling Research Domains," 2007.**

**[35]** **J. B. Luis Ribeiro, Gonçalo Cândido and Mauro Onori, "EVOLVABLE PRODUCTION SYSTEMS AN INTEGRATED VIEW ON RECENT**

**DEVEOLPMENTS," presented at the 6th International Conference on Digital Enterprise Technology, Hong Kong, 2009.**

**[36] D. Semere, *et al.*, "Evolvable Systems: Developments and Advance," p. 288–293.**

**[37] M. Onori, *et al.*, "Evolvable Assembly Systems: From Evaluation to Application," *Innovation in Manufacturing Networks*, pp. 205-214, 2008.**

**[38] R. Frei, *et al.*, "A complexity theory approach to evolvable production systems," 2007, pp. 44-53.**

**[39] J. Barata, *et al.*, "Diagnosis on evolvable production systems," 2007, pp. 3221-3226.**

**[40] R. Frei, *et al.*, "Evolvable assembly systems: Towards user friendly manufacturing," 2007, pp. 288-293.**

**[41] R. Isermann, *Fault-diagnosis systems*: Springer, 2006.**

**[42] C. Bocaniala and V. Palade, "Computational Intelligence Methodologies in Fault Diagnosis: Review and State of the Art," *Computational Intelligence in Fault Diagnosis*, pp. 1-36.**

**[43] I. Sommerville, "Critical Systems," S. Engeneering, Ed., 7th ed, 2004.**

**[44] R. Isermann, "Supervision, fault-detection and fault-diagnosis methods--An introduction," *Control engineering practice*, vol. 5, pp. 639-652, 1997.**

**[45] M. D. d. A. Krishna Nadiminti, and Rajkumar Buyya, "Distributed Systems and Recent Innovations: Challenges and Benefits."**

**[46] S. Dash and V. Venkatasubramanian, "Challenges in the industrial applications of fault diagnostic systems," *Computers & Chemical Engineering*, vol. 24, pp. 785-791, 2000.**

**[47] R. Reiter, "A theory of diagnosis from first principles," *Artificial Intelligence*, vol. 32, pp. 57-95, 1987.**

**[48] R. Milne, "Strategies for diagnosis," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 17, pp. 333-339, 1987.**

**[49] M. B. McBeth, *et al.*, "A generic strategy for diagnostic assistance: the technician's assistant," presented at the Proceedings of the 1st international conference on Industrial and engineering applications of artificial intelligence and expert systems - Volume 2, Tullahoma, Tennessee, United States, 1988.**

**[50] R. Isermann and P. Balle, "Trends in the application of model-based fault detection and diagnosis of technical processes," *Control Engineering Practice*, vol. 5, pp. 709-719, 1997.**

**[51] V. Venkatasubramanian, *et al.*, "A review of process fault detection and diagnosis:: Part II: Qualitative models and search strategies," *Computers & Chemical Engineering*, vol. 27, pp. 313-326, 2003.**

**[52] V. Venkatasubramanian, *et al.*, "A review of process fault detection and diagnosis:: Part III: Process history based methods," *Computers & Chemical Engineering*, vol. 27, pp. 327-346, 2003.**

**[53] R. Patton, *et al.*, "Artificial intelligence approaches to fault diagnosis," 1999.**

**[54] J. Gertler, "Analytical redundancy methods in fault detection and isolation," 1991, pp. 9–21.**

**[55] J. Gertler, "Fault detection and isolation using parity relations," *Control Engineering Practice*, vol. 5, pp. 653-661, 1997.**

**[56]** **A. Naik, Yin, Shen, Ding, Steven X., "Recursive Identification Algorithm for Parity Space Based Fault Detection Systems," presented at the 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, Sants Hotel, Spain, 2009.**

**[57]** **A. S. N. a. S. X. D. Shen Yin, "Adaptive process monitoring based on parity space methods," presented at the 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, Barcelona, Spain, 2009.**

**[58]** **G. Welch and G. Bishop, "An introduction to the Kalman filter," *University of North Carolina at Chapel Hill, Chapel Hill, NC,* 1995.**

**[59]** **Y. Wang and Y. Zheng, "Kalman filter based fault diagnosis of networked control system with white noise," *Journal of Control Theory and Applications,* vol. 3, pp. 55-59, 2005.**

**[60]** **L. An and N. Sepehri, "Hydraulic actuator circuit fault detection using extended kalman filter," 2003, pp. 4261-4266.**

**[61]** **H. Vedam and V. Venkatasubramanian, "Signed digraph based multiple fault diagnosis," *Computers & Chemical Engineering,* vol. 21, pp. S655-S660, 1997.**

**[62]** **M. Bhushan and R. Rengaswamy, "Design of sensor network based on the signed directed graph of the process for efficient fault diagnosis," *Ind. Eng. Chem. Res,* vol. 39, pp. 999-1019, 2000.**

**[63]** **S. Lapp, "Computer-aided synthesis of fault-trees," *IEEE Transactions on Reliability,* 1977.**

**[64]** **K. Forbus, "Qualitative reasoning," *CRC Handbook of Computer Science and Engineering,* pp. 715-733, 1996.**

**[65]** **S. LeClair and F. Abrams, "Qualitative process automation," *International Journal of Computer Integrated Manufacturing,* vol. 2, pp. 205-211, 1989.**

**[66]** **E. Segal, *et al.*, "Probabilistic abstraction hierarchies," *Advances in Neural Information Processing Systems,* vol. 2, pp. 913-920, 2002.**

**[67]** **E. Castillo, *et al.*, *Expert systems and probabilistic network models*: Springer Verlag, 1997.**

**[68]** **J. Reggia, *et al.*, "Diagnostic expert systems based on a set covering model," *International Journal of Man-Machine Studies,* vol. 19, pp. 437-460, 1983.**

**[69]** **P. Fink and J. Lusth, "Expert systems and diagnostic expertise in the mechanical and electrical domains," *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 17, pp. 340-349, 1987.**

**[70]** **M. Maurya, *et al.*, "A signed directed graph and qualitative trend analysis-based framework for incipient fault diagnosis," *Chemical Engineering Research and Design,* vol. 85, pp. 1407-1422, 2007.**

**[71]** **E. M. E. A. Baligh Mnassri, Bouchra Ananou, Mustapha Ouladsine, "Fault Detection and Diagnosis Based on PCA and a New Contribution Plots," presented at the 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, Barcelona, Spain, 2009.**

**[72]** **M. R. Joaquim Meléndez, Joan Colomer and Cesar Barta, "Statistical modelling for fault diagnosis of sensors of the Ariane's engine," presented at the 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, Barcelona, Spain, 2009.**

**[73]** **S. Haykin, *Neural networks: a comprehensive foundation*: Prentice Hall PTR Upper Saddle River, NJ, USA, 1994.**

**[74]** **Y. Maki and K. Loparo, "A neural-network approach to fault detection and diagnosis in industrial processes," *IEEE Transactions on control systems technology*, vol. 5, pp. 529-541, 1997.**

**[75]** **V. Venkatasubramanian and S. Rich, "An object-oriented two-tier architecture for integrating compiled and deep-level knowledge for process diagnosis," *Computers & Chemical Engineering*, vol. 12, pp. 903-921, 1988.**

**[76]** **M. Hofbaur and B. Williams, "Hybrid diagnosis with unknown behavioral modes," ed: Citeseer, 2002.**

**[77]** **F. Heck, *et al.*, "A multi-agent based monitoring and diagnosis system for industrial components," 1998, pp. 63–69.**

**[78]** **N. Roos, *et al.*, "A protocol for multi-agent diagnosis with spatially distributed knowledge," 2003, pp. 655-661.**

**[79]** **W. Gefang, *et al.*, "Research on Intellectualized Fault Diagnosis System Based on Distributed Multi-Agent Technology," 2007, pp. 3-405.**

**[80]** **M. Long, *et al.*, "Distributed multi-agent diagnosis and recovery from sensor failures," 2003, pp. 2506-2513.**

**[81]** **B. Horling, *et al.*, "Diagnosis as an integral part of multi-agent adaptability," 2000, p. 1211.**

**[82]** **J. Barata, *et al.*, "Diagnosis using Service Oriented Architectures (SOA)," 2007.**

**[83]** **L. Ribeiro, "A Diagnostic Infrastructure for Manufacturing Systems," 2007.**

**[84]** **X. Wu, *et al.*, "Web-based remote monitoring and fault diagnosis system," *The International Journal of Advanced Manufacturing Technology*, vol. 28, pp. 162-175, 2006.**

**[85]** **J. Son, *et al.*, "An integrated knowledge representation scheme and query processing mechanism for fault diagnosis in heterogeneous manufacturing environments," *Robotics and Computer-Integrated Manufacturing*, vol. 16, pp. 133-141, 2000.**

**[86]** **F. Balduzzi and A. Di Febbraro, "Combining fault detection and process optimization in manufacturing systems using first-order hybrid Petri nets," 2001, pp. 40-45.**

**[87]** **W. Hu, *et al.*, "Operational fault diagnosis of manufacturing systems," *Journal of Materials Processing Technology*, vol. 133, pp. 108-117, 2003.**

**[88]** **W. Hu, *et al.*, "A systematic approach to integrated fault diagnosis of flexible manufacturing systems," *International Journal of Machine Tools and Manufacture*, vol. 40, pp. 1587-1602, 2000.**

**[89]** **C. Cardenas, *et al.*, "Modelling, supervision and diagnosis of a manufacturing cell," 2003, pp. 69-74.**

**[90]** **M. McIntyre, *et al.*, "Fault identification for robot manipulators," *IEEE Transactions on Robotics*, vol. 21, pp. 1028-1034, 2005.**

**[91]** **K. K. N. Meskin, C. A. Rabbath, "Fault Consensus in a Network of Unmanned Vehicles," presented at the 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, Barcelona, Spain, 2009.**

**[92]** **J. Baker, "The DRAGON system--An overview," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 23, pp. 24-29, 1975.**

**[93]** **L. Rabiner, "A tutorial on hidden Markov models and selected applications inspeech recognition," *Proceedings of the IEEE*, vol. 77, pp. 257-286, 1989.**

**[94]** **B. Juang and L. Rabiner, "Hidden Markov models for speech recognition," *Technometrics*, vol. 33, pp. 251-272, 1991.**

**[95]** **S. Roweis, "Hidden Markov Models," in *SCIA 2003 Tutorial*, ed. Toronto, 2003.**

**[96]** **J. B. a. J. F. Luis Ribeiro, "A co-Evolving Diagnostic Algorithm for Evolvable Production Systems: A Case of Learning," 2010.**

**[97]** **P. Blunsom, "Hidden Markov Models," *The University of Melbourne, Department of Computer Science and Software Engineering, 19th August*, 2004.**

**[98]** **A. Daidone, *et al.*, "Hidden Markov models as a support for diagnosis: Formalization of the problem and synthesis of the solution," 2006, pp. 245-256.**

**[99]** **J. Ying, *et al.*, "A hidden Markov model-based algorithm for fault diagnosis with partial and imperfect tests," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 30, pp. 463-473, 2000.**

**[100]** **S. Dorogovtsev and J. Mendes, "Evolution of networks," *Advances in Physics*, vol. 51, pp. 1079-1187, 2002.**

**[101]** **N. Hall and S. Preiser, "Combined network complexity measures," *IBM Journal of Research and Development*, vol. 28, pp. 15-27, 1984.**

**[102]** **D. Bonchev and G. Buck, "Quantitative measures of network complexity," *Complexity in Chemistry, Biology, and Ecology*, pp. 191-235, 2005.**

**[103]** **J. Barata, *et al.*, "A Multiagent Based Control System Applied to an Educational Shop Floor," *Information Technology For Balanced Manufacturing Systems*, pp. 119-128, 2006.**

**[104]** **L. Ribeiro, *et al.*, "MAS and SOA: A Case Study Exploring Principles and Technologies to Support Self-Properties in Assembly Systems," 2008, pp. 192-197.**

**[105]** **L. Ribeiro, *et al.*, "An architecture for a fault tolerant highly reconfigurable shop floor," 2008, pp. 1194-1199.**

**[106]** **G. Cândido and J. Barata, "A Multiagent Control System for Shop Floor Assembly," *Holonic and Multi-Agent Systems for Manufacturing*, pp. 293-302, 2007.**

**[107]** **J. B. a. J. F. Luis Ribeiro, "An Agent-Based Interaction-Oriented Shop Floor to Support Emergent Diagnosis," presented at the 8th IEEE International Conference on Industrial Informatics, Osaka, Japan, 2010.**

**[108]** **M. Luck, *et al.*, "Agent technology: computing as interaction (a roadmap for agent based computing)," 2005.**

**[109]** **S. Bullock and D. Cliff, "Complexity and emergent behaviour in ICT systems," 2004.**

**[110]** **G. Huang, *et al.*, "Agent-based workflow management in collaborative product development on the Internet," *Computer Aided Design*, vol. 32, pp. 133-144, 2000.**

**[111]** **L. Ribeiro, *et al.*, "A generic communication interface for DPWS-based web services," 2008, pp. 762-767.**

**[112]** **D. Watts, "A simple model of global cascades on random networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, p. 5766, 2002.**

**[113]** **L. Ribeiro, *et al.*, "The Meaningfulness of Consensus and Context in Diagnosing Evolvable Production Systems," *Emerging Trends in Technological Innovation*, pp. 143-150, 2010.**

**[114]** **M. Genesereth and S. Ketchpel, "Software agents," *Commun. ACM*, vol. 37, pp. 48-53, 147, 1994.**

**[115]** **L. Ribeiro, "A co-Evolving Diagnostic Algorithm for Evolvable Production Systems: A Case of Learning," 2010.**

**[116]** **jm.francois. *jahmm - An implementation of Hidden Markov Models in Java*. Available: http://code.google.com/p/jahmm/**

**[117]** **T. Dunning, "Accurate methods for the statistics of surprise and coincidence," *Computational linguistics*, vol. 19, p. 74, 1993.**

**[118]** **J. B. L. Ribeiro, and J. Ferreira, "Emergent Diagnosis for Evolvable Production Systems," presented at the IEEE International Symposium on Industrial Electronics, Bari, Italy, 2010.**

**[119]** **L. Ribeiro, *et al.*, "Global vs Local: A Comparison of two Approaches to Perform Diagnosis in Networks of Mechatronic Agents."**

# 7 APPENDICES

## 7.1 Appendix 1 – Conveyer

```
er<?xml version="1.0" encoding="UTF-8"?>

<!--
    Document   : Settings.xml
    Created on : 12 de Janeiro de 2010, 13:21
    Author     : joaopsf
    Description:
        Purpose of the document follows.
-->

<Skills>
    <AgentName>Conv1</AgentName>
    <AgentType>C</AgentType>
    <AgentReference>Conv</AgentReference>
    <SkillList>
        <ComplexSkill>
            <Name>MoveLift</Name>
            <AID>to be defined</AID>
            <ID>1</ID>
            <isInstantitated>false</isInstantitated>
            <Arguments>
            </Arguments>
            <CompositionGraph>
                <AtomicSkill>
                    <Name>to be defined</Name>
                    <ID>0</ID>
                    <AID>to be defined</AID>
                    <isInstantitated>false</isInstantitated>
                    <Arguments>
                        <Argument>
                            <Name>Actuate</Name>
                            <Value>0.0</Value>
                            <Offset>0.0</Offset>
                        </Argument>
                    </Arguments>
                    <TimeSlot>0</TimeSlot>
                    <Previous>0</Previous>
                    <RequiredModule>C</RequiredModule>
                </AtomicSkill>
```

```
                <AtomicSkill>
                    <Name>to be defined</Name>
                    <ID>0</ID>
                    <AID>to be defined</AID>
                    <isInstantitated>false</isInstantitated>
                    <Arguments>
                        <Argument>
                            <Name>Actuate</Name>
                            <Value>0.0</Value>
                            <Offset>0.0</Offset>
                        </Argument>
                    </Arguments>
                    <TimeSlot>0</TimeSlot>
                    <Previous>0</Previous>
                    <RequiredModule>C</RequiredModule>
                </AtomicSkill>
            </CompositionGraph>
        </ComplexSkill>
        <Restrictions>
            <Restriction>
                <ElementType>C</ElementType>
                <RelationType>F</RelationType>
                <ReferenceType>WorkflowInstantiation</ReferenceType>
            </Restriction>
        </Restrictions>
        <isInstantitated>false</isInstantitated>
        <Nature>ComplexSkill</Nature>
        <RequiredModules>C</RequiredModules>
    </SkillList>
    <SkillList>
        <SimpleSkill>
            <Name>Conv10</Name>
            <ID>2</ID>
            <AID>to be defined</AID>
            <isInstantitated>false</isInstantitated>
            <Arguments>
                <Argument>
                    <Name>Actuate</Name>
                    <Value>0.0</Value>
                    <Offset>0.0</Offset>
                </Argument>
            </Arguments>
            <TimeSlot>1</TimeSlot>
            <Previous>0</Previous>
            <RequiredModule>C</RequiredModule>
        </SimpleSkill>
        <Restrictions>
            <Restriction>
                <ElementType>C</ElementType>
                <RelationType>F</RelationType>
                <ReferenceType>WorkflowInstantiation</ReferenceType>
            </Restriction>
        </Restrictions>
        <isInstantitated>false</isInstantitated>
        <Nature>SimpleSkill</Nature>
        <RequiredModules>C</RequiredModules>
    </SkillList>
```

```
<AlternativeSkillList>
    <ComplexSkill>
        <Name>MoveLift</Name>
        <AID>to be defined</AID>
        <ID>1</ID>
        <isInstantitated>false</isInstantitated>
        <Arguments>
        </Arguments>
        <CompositionGraph>
            <AtomicSkill>
                <Name>Conv10</Name>
                <ID>1</ID>
                <AlternativeSkill>6</AlternativeSkill>
                <AID>to be defined</AID>
                <isInstantitated>false</isInstantitated>
                <Arguments>
                    <Argument>
                        <Name>Actuate</Name>
                        <Value>0.0</Value>
                        <Offset>0.0</Offset>
                    </Argument>
                </Arguments>
                <TimeSlot>1</TimeSlot>
                <Previous>0</Previous>
                <RequiredModule>C</RequiredModule>
            </AtomicSkill>
            <AtomicSkill>
                <Name>Lift4</Name>
                <ID>2</ID>
                <AlternativeSkill>6</AlternativeSkill>
                <AID>to be defined</AID>
                <isInstantitated>false</isInstantitated>
                <Arguments>
                    <Argument>
                        <Name>Actuate</Name>
                        <Value>0.0</Value>
                        <Offset>0.0</Offset>
                    </Argument>
                </Arguments>
                <TimeSlot>2</TimeSlot>
                <Previous>1</Previous>
                <RequiredModule>C</RequiredModule>
            </AtomicSkill>
        </CompositionGraph>
    </ComplexSkill>
    <Restrictions>
        <Restriction>
            <ElementType>C</ElementType>
            <RelationType>F</RelationType>
            <ReferenceType>WorkflowInstantiation</ReferenceType>
        </Restriction>
    </Restrictions>
    <isInstantitated>false</isInstantitated>
    <Nature>ComplexSkill</Nature>
    <RequiredModules>C</RequiredModules>
</AlternativeSkillList>
```

```
    <AlternativeSkillList>
        <SimpleSkill>
            <Name>Conv10</Name>
            <ID>2</ID>
            <AlternativeSkill>6</AlternativeSkill>
            <AID>to be defined</AID>
            <isInstantitated>false</isInstantitated>
            <Arguments>
                <Argument>
                    <Name>Actuate</Name>
                    <Value>0.0</Value>
                    <Offset>0.0</Offset>
                </Argument>
            </Arguments>
            <TimeSlot>1</TimeSlot>
            <Previous>0</Previous>
            <RequiredModule>C</RequiredModule>
        </SimpleSkill>
        <Restrictions>
            <Restriction>
                <ElementType>C</ElementType>
                <RelationType>F</RelationType>
                <ReferenceType>WorkflowInstantiation</ReferenceType>
            </Restriction>
        </Restrictions>
        <isInstantitated>false</isInstantitated>
        <Nature>SimpleSkill</Nature>
        <RequiredModules>C</RequiredModules>
    </AlternativeSkillList>
    <AMItype>AMI</AMItype>
    <AMIName>PLC_AMI</AMIName>
</Skills>
```